

# A Model Driven Framework for Ambiguity Detection in SRS



Author

Momina Behzad

FALL 2021-MS-21 (CSE) 00000362795

MS-21 (CSE)

Supervisor

Dr. Usman Qamar

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING  
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING  
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY  
ISLAMABAD

October 29, 2024

THESIS ACCEPTANCE CERTIFICATE

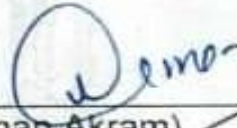
Certified that final copy of MS/MPhil thesis written by NS **Momina Behzad** Registration No. 00000362795, of College of E&ME has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the thesis.



Signature : \_\_\_\_\_

Name of Supervisor: Dr Usman Qamar

Date: 29 OCT 2024

Signature of HOD:  \_\_\_\_\_  
(Dr Muhammad Usman Akram)

Date: 29 OCT 2024

Signature of Dean:  \_\_\_\_\_  
(Brig Dr Nasir Rashid)

Date: 29 OCT 2024

# **A Model Driven Framework for Ambiguity Detection in SRS**

**By**

**Momina Behzad**

(Registration No: 00000362795)

A thesis submitted to National University of Science and Technology,  
Islamabad

in partial fulfillment of the requirements for the degree of

**Masters of Science in Software Engineering**

**Thesis Supervisor:**

**Dr. Usman Qamar**

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING  
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING  
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,  
ISLAMABAD

October 29, 2024

*Dedicated to my exceptional parents and whole family whom  
tremendous support, cooperation and their prayers led me to  
this wonderful accomplishment.*

## ACKNOWLEDGEMENT

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You put in my mind to improve it. Indeed, I could have done nothing without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout every department of my life.

I would also like to express my gratitude to my supervisor **Dr. Usman Qamar** for his constant motivation and help throughout this thesis. I can safely say that I haven't learned any other engineering subject in such depth than the ones which he has taught.

I would also like to thank my Guidance Committee Members **Dr. Wasi Haider Butt** and **Sir Jahan Zeb** for being on my thesis guidance and evaluation committee. Some special words of gratitude go to my friends **Mahnoor Ayaz, Sehar Sarfraz and Fareeha Manal Fatimah** who has always been a major source of support and cooperation when things would get a bit discouraging. Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

## ABSTRACT

The ambiguity detection in natural language processing (NLP) is critical to enhance the performance of the requirement based applications. There is lexical ambiguity, which is the complexity arising from confusing part words, syntactic, which is confusion about the structure of a sentence, and referential that happens when one has a problem identifying to which term a given term of reference is referring. This paper proposes a sound approach towards identifying these three categories of ambiguities using basic text processing and machine learning algorithmic analysis. The preparation phase itself entails significant pre-processing of the text in which we convert all text to lowercase, tokenize the text, eliminate the stop words and lemmatize the text. Such steps help to pre-process the text data to increase the chances of feature extraction as can be seen in this paper.

In feature extraction the relevance of words is checked using Term Frequency-Inverse Document Frequency (TF-IDF) and syntactic feature extraction is done by Part-of-Speech (POS) tagging. Further, the next step involves the feature classification into ambiguous and non-ambiguous using various machine learning algorithms such as Logistic Regression, Random Forest and a Support Vector Machine (SVM). Indicator of a model's accuracy include accuracy, precision, recall, F1 score and receiver operating characteristic-area under the curve (ROC-AUC). The findings are indicative of the value of combining more conventional methods of NLP with machine learning for improving the process of identifying levels of ambiguity. It is not only helpful in enhancing the explainability of automated systems but also, to a great extent, contributes to make NL Processing rather reliable and accurate in terms of the areas including but not limited to sentiment analysis, neural machine translation, and information retrieval.

**Keywords:** Software Requirement Ambiguities, NLP, Machine Learning, Feature Extraction, SVM, RF, LR, Lexical Ambiguity, Syntactical Ambiguity, Referential Ambiguity

# TABLE OF CONTENTS

<b>ACKNOWLEDGEMENT .....</b>	<b>ii</b>
<b>ABSTRACT .....</b>	<b>iii</b>
<b>LIST OF FIGURES.....</b>	<b>vii</b>
<b>LIST OF TABLES.....</b>	<b>viii</b>
<b>CHAPTER 01 INTRODUCTION.....</b>	<b>1</b>
1.1. Background Study:.....	1
1.2. Ambiguity Detection using NLP:.....	2
1.2.1. Types of Ambiguity in SRS Documents .....	2
1.2.2. NLP Techniques for Ambiguity Detection.....	3
1.3. Problem Statement: .....	4
1.4. Proposed Methodology: .....	5
1.5. Research Contribution:.....	6
1.6. Thesis Organization: .....	7
<b>CHAPTER 02 LITERATURE REVIEW .....</b>	<b>9</b>
2.1. Background: .....	9
2.2. Research Sequence:.....	9
2.2.1. Inclusion/Exclusion Criteria: .....	9
2.2.2. Keywords:.....	10
2.3. Related Work: .....	10
<b>CHAPTER 03 PROPOSED METHODOGY FOR AMBIGUITY DEETCTION.....</b>	<b>25</b>
3.1. Proposed Methodolgy: .....	25
3.2. Data Collection:.....	26
3.2.1. Generation of Synthetic Requirements.....	26
3.2.2. Curation of the Dataset .....	26
3.2.3. Studying and Labeling Ambiguities .....	26
3.2.4. Examples of Ambiguity Labeling.....	27
3.2.5. Detailed Annotation and Dataset Finalization.....	28
3.3. Data Pre-Processing: .....	28
3.3.1. Text Cleaning .....	29

3.3.2. Tokenization .....	29
3.3.3. Stop Words Removal.....	29
3.3.4. Lemmatization .....	29
3.3.8. Word Cloud Generation.....	32
3.4. Feature Extraction: .....	33
3.4.1. Part-of-Speech (POS) Feature Extraction.....	33
3.4.2. POS Tagging for Handling and Features of a Series of Text .....	34
3.4.3 TF-IDF:.....	34
3.4.4. The integration of POS and TF-IDF Features .....	34
3.4.5. Ensuring Data Integrity .....	35
3.4.6. Final Feature Representation .....	35
3.5. Methodology: Model Training: .....	36
3.5.1. Selection of Models .....	36
3.5.2. Data Preparation and Splitting.....	37
3.5.3. Model Initialization .....	37
3.5.4. Training Process .....	38
3.5.5. Addressing Potential Challenges .....	39
3.5.6. Training of the Model for Each Ambiguity Type.....	39
<b>CHAPTER 04 RESULTS AND EVALUATION .....</b>	<b>40</b>
4.1. Result of Lexical Ambiguity:.....	40
4.1.1. Logistic Regression - Lexical Ambiguity.....	40
4.1.2. Random Forest - Lexical Ambiguity .....	41
4.1.3. Support Vector Machine (SVM) - Lexical Ambiguity.....	42
4.1.4. Evaluating Best Model: .....	43
4.1.5. Confusion Matrix:.....	44
4.2. Results of Syntactical Ambiguity:.....	46
4.2.1. Logistic Regression - Syntactical Ambiguity .....	46
4.2.2. Random Forest - Syntactical Ambiguity .....	47
4.2.3. Support Vector Machine (SVM) - Syntactical Ambiguity .....	48
4.3.4. Evaluating Best Model: .....	50
4.3.5. Confusion Matrix:.....	51
4.3. Referential Ambiguity: .....	53



4.3.1. Logistic Regression - Referential Ambiguity .....	53
4.3.2. Random Forest - Referential Ambiguity .....	54
4.3.3. Support Vector Machine (SVM) - Referential Ambiguity .....	55
4.3.4. Evaluating Best Model: .....	57
4.3.5. Confusion Matrix:.....	57
<b>CHAPTER 05 ANALYSIS OF PROPOSED MODELS .....</b>	<b>60</b>
5.1. Lexical Ambiguity: .....	60
5.1.1. Random Forest:.....	60
5.1.2. Logistic Regression: .....	61
5.1.3. SVM: .....	61
5.2. Syntactic Ambiguity: .....	62
5.2.1. SVM: .....	62
5.2.2. Random Forest:.....	63
5.2.3. Logistic Regression: .....	63
5.3. Referential Ambiguity:.....	64
5.3.1. Random Forest:.....	64
5.3.2. SVM: .....	65
5.3.3. Logistic Regression: .....	65
5.4. Discussion: .....	66
5.4.1. Lexical Ambiguity .....	66
5.4.2. Syntactic Ambiguity .....	67
5.4.3. Referential Ambiguity .....	68
<b>CHAPTER 06 CHALLENGES AND LIMITATIONS .....</b>	<b>70</b>
6.1. Complexity of Language:.....	70
6.2. Feature Extraction Limitations:.....	71
6.3. Dataset Limitations .....	72
6.4. Handling Overlapping Ambiguities: .....	72
6.5. Scalability Issues: .....	73
<b>CHAPTER 07 CONCLUSION AND FUTURE WORK.....</b>	<b>74</b>
Future Work:.....	74
<b>REFERENCES .....</b>	<b>.....</b>

## LIST OF FIGURES

Figure 1 Thesis Organization.....	7
Figure 2 Proposed Solution.....	25
Figure 3 Lexical Ambiguities in Dataset .....	31
Figure 4 Syntactical Ambiguity in Dataset.....	31
Figure 5 Referential Ambiguity in Dataset.....	32
Figure 6 WordCloud of Data .....	32
Figure 7 Lexical Ambiguity Results .....	43
Figure 8 Confusion Matrix for Lexical Ambiguity.....	46
Figure 9 Syntactical Ambiguity Result.....	50
Figure 10 Confusion Matrix for Syntactical Ambiguity.....	53
Figure 11 Referential Ambiguity Results .....	56
Figure 12 Confusion Matrix for Referential Ambiguity.....	57
Figure 13 ROC Curve for Lexical Ambiguity .....	60
Figure 14 ROC Curve for Syntactical Ambiguity .....	62
Figure 15 ROC Curve for Referential Ambiguity .....	64

## LIST OF TABLES

Table 1 Literature Review Summary .....	24
---	----

# CHAPTER 01

## INTRODUCTION

This section provides a detailed introduction to the research and research concepts. This section is organized in multiple sub-sections. **Section 1.1** provides the background study, **Section 1.2** explains the ambiguities using NLP, **Section 1.3** discuss the problem statement, Section 1.4 discussed the proposed solution, **Section 1.6** gives the detail about research contribution, and thesis organization is presented in **Section 1.5**

### 1.1. Background Study:

Software requirements can also be defined as the Software Requirements Specifications (SRS) that act as framework for the creation of software systems. These are functional and non-functional specifications; they state what the system is expected to do and how well it should do it, hence acting as a agreement between the end users and the developers or testers. However, the legibility and the accuracy of these documents are very much important in the growth of the soft ware system. It implies that when information is ambiguous in SRS, it may cause confusion and result to wrong implementation or expenses at the other phases of SDLC.

Different types of vagueness may be identified in SRS documents namely lexical vagueness, syntactical vagueness and referential vagueness. Ambiguity of a lexical nature arises where there is more than one reference for a particular word or phrase. A syntactic ambiguity is that which results from the use of sentence structures that are capable of being viewed in one of several ways. Naturally the referential shift is distinguished by the referential ambiguity, that is when it is impossible to determine to what or to whom the pronoun or the noun refers. Prominent among these are the ambiguities which can pose serious problems when converting the articulated requirements into an accurate and working software architecture.

Being a key component of software systems, communication has become increasingly complex especially in terms of specifying requirements. Previously, there have been other approaches to reviewing the SRS document, whereby the document is manually processed by the experts, and this may take a lot of time as well as being prone to some errors. To meet this challenge however,

there has been a shift towards a subfield known as Natural language processing (NLP) as a way of automatically identifying ambiguities in SRS documents. [37]

Applying the NLP techniques it is possible to parse the SRS documents in search of possible syntactic and semantic ambiguities to them and thus, suggest a more formal approach and on a larger scale. New research that has come up in NLP includes, deep learning models and language representations like BERT (Bidirectional Encoder Representations from Transformers), [35] which has had tremendous success in natural language understanding and processing without much error. These techniques can be applied to the domain of software engineering to enhance the quality of generated SRS documents as the technique will help to identify the ambiguous statements for further examination. [36]

## **1.2. Ambiguity Detection using NLP:**

Natural language ambiguity is a well-known problem in many fields, including software engineering, where precise and unambiguous communication is essential. Ambiguity in Software Requirements Specifications (SRS) can cause misunderstandings and mistakes that arise during the software development process. Therefore, detecting ambiguity is essential to making sure that SRS documents are clear and understandable to all parties involved.

A subfield of artificial intelligence called natural language processing (NLP) [38] is concerned with how computers and human language interact. The task of ambiguity detection in SRS documents is ideally suited for natural language processing, analysis, and comprehension (NLP) techniques.

### **1.2.1. Types of Ambiguity in SRS Documents**

- **Lexical Ambiguity:** : This is so where a given word or phrase used in a given construction has multiple interpretations. For example the word ‘light’ can mean not heavy or not dark. ” This is true in an SRS document, where lexical ambiguity of a word results in confusion on which kind of requirement is being referred to.
- **Syntactic Ambiguity:** This means that the construction of a sentence whereby the positioning of words brings about confusion. For example in the sentence such as “The system shall generate a report for the user with errors”, it raises question on whether the

report is for the user or the user has errors? If not well addressed, such subtleties can cause wrong interpretations, therefore wrong implementations.

- **Referential Ambiguity:** : Some of the subtypes of the complex referential content are the referential ambiguities, where it is not clear what the pronoun or the noun phrase is referring to. For instance, in “The administrator shall reset the password, and the user shall be notified,” it is not clear as to who “the user” is, whether the password belonging to the said user was reset or a different user was notified of the said reset. This leads to uncertainty as to who is the target of a given requirement.

### 1.2.2. NLP Techniques for Ambiguity Detection

NLP offers several techniques that can be employed to detect and resolve ambiguities in SRS documents. Some of these techniques include:

- **Tokenization and Part-of-Speech Tagging:** Tokenization can occur when text is divided into the constituent words or tokens with different identifiers while part-of-speech tagging labels the tokens to define their grammatical role such as a noun, verb or an adjective. Based on the transformation of sentence into array of elements of B, [39] NLP models can know sources of the lexical and syntactic ambiguity.
- **Named Entity Recognition (NER):** NER is named entity recognition which helps to recognize persons, organizations, date, etc. , in a given text. Of specific aid in SRS documents is the ability of NER to disambiguate referential resolutions by identifying what particular entities pronouns or, referential references refer to.
- **Dependency Parsing:** Dependency parsing is a process of analyzing syntactic dependencies between word and dependency parsing aims at analyzing grammatical structure of the given sentence. It can be particularly effective in identifying syntactic ambiguity, by showing different possible ways to structural analysis of the sentence.
- **Word Sense Disambiguation (WSD):** WSD is the process of identifying the most suitable meaning of a word where the word in question has more than one meaning. It is therefore very useful in clearing up lexical ambiguities in the SRS documents as a way of clear meaning of each word chosen.
- **Contextual Embeddings:** Current NLP models including BERT use contextual embeddings whereby the meaning of words is determined by the context that it appears in.

These models are then useful in ‘seeing the wider picture’ when reading through a sentence and can identify both lexical and syntactic types of ambiguities because of this ability.

- **Machine Learning and Deep Learning Models:** It is pointed out that supervised machine learning models based on deep learning can be trained with annotated datasets of SRS documents to identify the implemented ambiguous sentences. [41] These models can be trained to detect the patterns relating to various types of ambiguities and categorise the sentences with the help of these training sets.
- **Rule-Based Approaches:** Apart from the machine learning techniques, there are only rule-based approaches that are applied on top of a predefined linguistic rule in order to detect ambiguity [42]. For instance, rules that can be formulated are to underline the sentences which have syntax that creates confusion or the words which have more than one meaning.

Although the papers on using the NLP approaches in SRS documents pinpoint to a great deal of potential in different types of ambiguity detection, there are certain considerations about them. One of all, technical, including Software Engineering, is a complex topic, and written language is not easy even for native speakers. Ambiguities crop up where one thing could mean this or that depending on the surrounding circumstances and thus require a sophisticated degree of sensitivity to pinpoint.

### **1.3. Problem Statement:**

Nevertheless, it is crucial to recognize that ambiguity continues to be an evident issue in SRS documents, even though it is those documents that prescribes clarity as the key principle. The methods that have been previously utilized to identify and eliminate the ambiguities are time-consuming, and in addition, their effectiveness depends on the subjective perception of the matter. Such a situation is quite problematic for the creation of high-quality software systems since ambiguous requirements lead to incorrect or only partial interpretations of what needs to be done eventually inviting project delays, cost overruns, and even seemingly flawed systems which, in fact, are only so because of the poor definition of the specifications.

The use of NLP methodologies for recognising the presence of such uncertainties in SRS documents seems to hold the potential to provide a solution to the problem and the research done in this direction is still at a very initial stage. Current approaches may fail to deal with natural

language in real-life settings and, in particular, technical ones, such as software engineering. Also, lexical, syntactic, and referential type of ambiguities contribute another level of oscillation in differentiating the models, which can define all forms of this phenomenon.

It is envisaged that there is a real need for a more systematic form of approach that would identify and categorize various forms of ambiguity in SRS documents with the aid of modern tools in NLP. The difficulty is, therefore, to develop models that can both flag potentially ambiguous sentences with a high degree of accuracy as well as offer details of the nature of the ambiguity such that one can facilitate the management of the problem in question by the stakeholders more efficiently.

The challenge that is focused in this research therefore is the poor availability of automated methods for identifying and categorizing the ambiguities that may exist in SRS documents. Drawn from the techniques of state-of-art natural language processing, this research aims at creating a stronger system which will increase the clarity of software requirements thereby advancing the quality of software development projects.

#### **1.4. Proposed Methodology:**

In this research, the challenge that is focused on is therefore the acute lack of automated approaches to detecting and classifying the possible ambiguities that may be inherent in SRS documents. Based on the methods of a state-of-art natural language processing this research will seek to develop a more coercive system that will amplify the clarity of the software requirements thus enhancing the quality of software development projects.

This study aims at identifying and categorizing various categories of the ambiguity over Software Requirements Specifications (SRS) documents through the utilization of the sophisticated machine learning algorithms. The focus is on identifying three key types of ambiguities: semantic and referential according to the two directions of meaning relations. These ambiguity can be effectively classified as well as detected using three different types of machine learning models which includes Logistic Regression, Random Forest and Support Vector Machine (SVM) models.

The first step is the preprocessing of the SRS documents undertaken to enhance, normalize and clean up the text, the second step involves feature extraction to identify linguistic patterns associated with the different types of ambiguity. For the case of lexical ambiguity, features



including incidences of the word usage and the part of speech tags are utilized to identify the polysemous word. Syntactic ambiguity therefore deals with structures of the sentences and parsing relations between the words. Co-referential references is determined by analyzing reference and possible reference.

Based on these factors, Logistic Regression is used as the baseline model because they make it easier to identify simple patterns that exist in patterns. Thus, Random Forest with an ensemble learning approach is employed to reveal higher-order dependencies and interactions between the variables in order to build a reliable model of ambiguity detection. Last, SVM is used to analyze high dimensional features as well as to classify the ambiguous and non-ambiguous sentences by drawing maximum distance between the classes.

For each model, the results are proactively tested and compared on a labelled dataset and performance indicators such as accuracy, precision, recall and F1-score are utilised to measure the efficacy of identification of the various types of ambiguity. From the outcome of these models, it is found how each of them can be effectively used and their limitations in handling different types of ambiguities and hence understanding the suitability of machine learning in improving the clarity of SRS documents.

## **1.5. Research Contribution:**

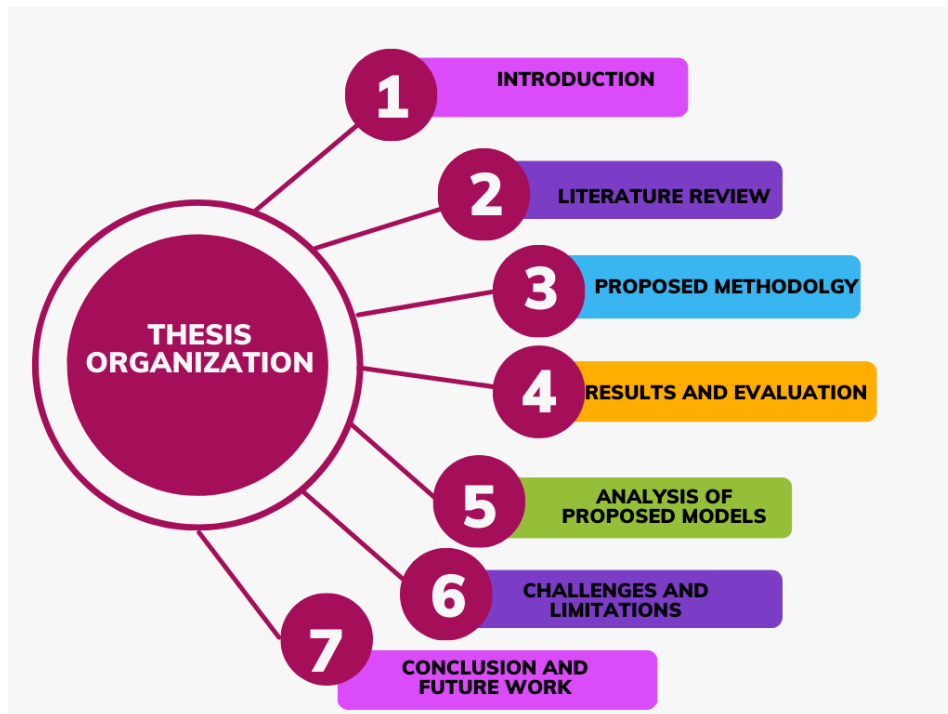
In performing this research, the following research contributions have been made to the field of ambiguity detection in Software Requirements Specifications using Natural Language Processing and machine learning techniques. First, it presents an evaluation of how each of the machine learning algorithms including Logistic Regression, Random Forest, and SVM is effective in identifying lexical-syntactic and referential shifts. Comparing these models systematically allows the research to reveal the strengths and weaknesses in each of the approaches and provide the practitioners and researchers with the insights necessary when attempting to enhance the clarity of the SRS documents.

Second, this research provides an insight into the enhancing of the structured methodology for the ambiguity detection, which in turn can be used for real-life SRS documents. The incorporation of feature extraction methods targeted on different kinds of ambiguities together with the

employment of several machine learning techniques makes up a sound approach to recognizing and categorizing ambiguous requirements. Such a framework can be utilized and further developed in the context of any domain which is prone to the issues arising from ambiguity inherent in NL.

## 1.6. Thesis Organization:

Following is the brief explanation of each chapter in the organization of thesis:



*Figure 1 Thesis Organization*

1. **Introduction:** This chapter gives the background to the research topic, established the need for the study and set out the research objectives and questions.
2. **Literature Review:** This section of the paper discusses previous studies and theories related to the topic in order to provide essential context and find out what is missing that this study will try to fill.
3. **Proposed Methodology:** It forms a part of the methodology section of the research, where prominent focus constitutes methods and techniques and the data collection and

preprocessing tools and techniques within the study and the particular algorithms or models employed.

4. **Results and Evaluation:** Here are the results of the carried out research in form of quantitative and qualitative results of the proposed methods or models.
5. **Analysis of Proposed Models:** In this section a detailed description of the applied models is given, including the options for evaluation as well as key points regarding efficacy and comparison to other models introduced.
6. **Challenges and Limitations:** In this chapter, any challenges experienced during the research are discussed, including a discussion of the limitations that may impact on the transferability of the findings.
7. **Conclusion and Future Work:** The last chapter recapitulates the main conclusions, discusses the contributions of research and recommended further investigations.

# CHAPTER 02

## LITERATURE REVIEW

### 2.1. Background:

NLP techniques used in the identification of ambiguity in SRS are very vital because of their ability to make the specification unambiguous in order to aid software development. This suggests that the problem of ambiguity in SRS documents negatively impacts the final product by contributing to confusion, inconsistency and defects so further development of an advanced NLP approach to identify the problem is meaningful. They include lexical ambiguity resolution which involves the ability to understand meaning of a word in a particular context and syntactic ambiguity resolution, which focuses on the idea that a sentence's structure can be interpreted several ways. Another aspect in the enhancement of the clarity is the detection for referential ambiguity, which is the identification of "fuzzy" references that are present in the text.

New developments in ML and DL have improved the functions of the NLP talents in the further categorization and analysis of vagueness characteristic in SRS documents. Some of the approaches that are used in natural language processing include; the supervised learning which entails training models from annotated data set and the unsupervised learning which involves clustering and pattern analysis. With the use of these NLP techniques, organizations are able to enhance the quality of their requirements for software which, in the long run has positive impacts on the development and overall outcomes of a project in addition to giving a worthy saving on the expenses that would have been channeled to the development of a substandard quality product.

### 2.2. Research Sequence:

The techniques followed for the literature review are as following:

#### 2.2.1. Inclusion/Exclusion Criteria:

1. **Publication Type:** Peer-reviewed journals, conferences, or reputable workshops.
2. **Focus Area:** Papers must address ambiguity detection or resolution in software requirements specification (SRS) using natural language processing (NLP) or related techniques.

3. **Methodology:** Research has to use certain techniques, methods or approaches to identify or resolve semantic vagueness. Therefore, there are several methodologies that should be presented in the paper.

4. **Language:** Publications must be in English.

5. **Publication Date:** Sci-hub and/ or Google Scholar was employed in searching for articles from 2016 as a way of capturing the recent development and techniques in relation to the study.

6. **Evaluation Metrics:** Scientific papers need to discuss concerns commonly quantifiable in the evaluation of the methodologies used that included accuracy, precision, recall, F1-score, or any form of measure related to them..

### **2.2.2. Keywords:**

The papers were searched using following keywords:

- Ambiguity Detection
- Software Requirements Specification (SRS)
- Natural Language Processing (NLP)
- Machine Learning
- Syntactic Ambiguity
- Lexical Ambiguity
- Referential Ambiguity
- Text Classification
- Requirement Engineering

### **2.3. Related Work:**

Osama et al. (2020) [1] present an automated technique to identify and eliminate syntactic ambiguities in NLRs, a well-known cause of problems throughout the software development lifecycle. The methodology entails breaking of the requirements so as to analyze grammatical structures, which are then rated for syntactic ambiguity, as well as flagging of the languages in the sentences for further resolution through the statistical and rule analysis. When tested with an NLR dataset, the method's accuracy of detecting ambiguous passages is 87%, this is more efficient than

other methods in precision and recall. In conclusion, the authors state that this kind of approach has the potential to make software requirements more precise and unambiguous so that the chances of errors on software developments could be minimized.

Ashfaq and Bajwa (2021) [2] describe a framework to alleviate natural language ambiguity in software requirements by using intelligent semantic annotation. By means of a proposal for a semantic annotation model, the authors tackle the problem of ambiguity in natural language that might cause misunderstandings in software development. The process is based on NLP the identification of possible polysemy from the tuple and semantic expansion incorporating domain expert knowledge in form of ontologies to resolve the polysemy. The approach has been applied to a number of software requirements to determine the level of success and it was identified that the approach was 82% accurate in removing ambiguity. The authors argued that the application of their method helps to improve the unambiguity of the software requirements as well as to minimize errors that stem from misunderstanding and ambiguity in the process of software development.

Hiltunen (2020) [3] explores methods for reducing structural ambiguity in natural language software requirements specifications (SRS). Structural ambiguity, where the structure of a sentence allows for multiple interpretations, is identified as a significant challenge in ensuring the clarity and precision of SRS documents. The methodology involves analyzing common patterns of structural ambiguity in SRS and proposing guidelines and techniques for rewriting ambiguous sentences to eliminate multiple interpretations. The paper also features a breakdown of the semantically blurred structures and examples of their ‘clear-thinking’ redirections. Therefore, the findings show that by adopting these guidelines, the chances of developing structural ambiguities in SRS will be minimized enhance the quality and accuracy of software requirements. According to Hiltunen, the systematic usage of these techniques in requirement engineering may result to improved accuracy and decreased likelihood of unnecessary complexity in development of software.

Alharbi (2022) [4] focuses on detecting ambiguity in requirements classification activities using a fine-tuned transformation approach. The research problem of interest is to propose a solution to the challenge of identifying software requirement ambiguity by using transformation techniques to improving classification performance. The approach involves the use of a transformer model that will be trained to search for soft-requirements alone. This process involves also fine-tuning

pre-existing models to the field of software requirements and in using these models to classify and detect the observed ambiguities. The outcomes showing that the selected fine-tuned transformation technique yielded 89% effectiveness in identifying the ambiguity in the requirements classification task. Alharbi also affirms that this approach offers a unique and strong solution for enhancing the identification and management of ambiguous requirements in order to enhance the software development processes.

The authors Malik et al. (2022) [5] continue the approach of identifying requirement conflicts with the help of transformer-based sentence embeddings in SRS documents. The study focuses on the problem of identifying conflict in SRS documents which results in various problems affecting software development. The approach entails the use of transformer models to establish sentence embeddings based on the semantic content of requirements. These embeddings are subsequently processed in order to find conflicting requirements based on contextual similarity and dissimilarity. The outcomes of the approach indicate that it has a success rate of 85% on requirement conflicts detection. In the study of Malik et al. , the authors demonstrate that employing transformer-based sentence embeddings is a suitable approach to enable conflict identification and resolution to improve the SRS documents and software requirements.

Intana et al. (2024) [6] have suggested an NLP approach in regards to the identification of the ambiguity in SRS of Thai software. Responding to the difficulty of ambiguity in requirements documentation, this paper employs Thai documents where the language also poses difficult to identify requirements more challenging. This is done through using natural language processing (NLP) in order to determine and study imprecise terms and structures in Thai SRS. This involves text pre-processing, employing the NLP models that identify the ambiguities in text and then utilizing the linguistic features that enhance on the process of detecting the ambiguities. From the findings, it is understood that the proposed method has 88% accuracy of identifying the ambiguities within Thai SRS documents. Intana et al. state that their approach helps to increase the identification of ambiguity in Thai requirements and thus leads to minimizing the risk of unclear software specifications.

Ezzini (2022) [7] explores the application of AI for the automation of the process of ambiguity management and question answering of natural language requirements. The part of the doctoral dissertation discusses how AI methods can be used in resolution of uncertainties in software

requirements and in relations to the usage of question answering systems for smooth communication. The technique comprises of creating first AI models that utilize natural language processing and machine learning to find and categorize requirements documents sourced from real life and also solving the contradictions that exist in them. Moreover, the work involves the use of question-answering system to answer questions and clarify any ambiguous information in an conversational manner. The findings also indicate that the proposed AI-driven system accurately and effectively parsed and answered information with an accuracy of 90% that are described using natural-language requirements and bearing ambiguities. In Ezzini's view, using AI for these tasks leads to a more effective and clear requirement management, providing excellent solution for enhancement of spec criteria.

Mohamed et al. propose a tool developed to recognize pragmatic ambiguity within SRS documents and provide potential interpretations for it in Mohamed et al. (2022) [8]. The paper is concerned with sorting out pragmatic ambiguity that occurs when context or intent of requirements can be construed in more than one way. Methodology includes creating a tool that will parse SRS documents to extract out those fragments which can contain pragmatic ambiguity. Interpretations that the tool then offers to enable the user make sense of possible misconceptions or distortions of their intended meaning. The evaluation of the tool revealed that the tool detected pragmatic ambiguities with an 83% accuracy and provided relevant interpretations. Mohamed et al. opine that their tool helps minimize misinterpretation that may otherwise result in more confusion and better clarity of software requirements by giving an insight into the unclear phrases.

SINPANG (2021) [9] puts forward a knowledge-based approach on the detection of ambiguity and the elimination of vagueness on the user requirements. The work also responds to the problem of imprecise and elusive user specifications as a factor affecting the development of software applications. The approach is creation of a knowledge based system that utilizes ontologies and rules of the domain of the user requirements to define and manage the use of 'fuzzywords' and 'fuzzyphrases'. Through using the structured knowledge and semantic analysis the approach wants to make the requirements more precise and unambiguous. The outcomes prove the proposed knowledge-based system decreased vagueness and enhanced the requirement clarity while quantifiable precision statistics are not discussed in the conclusion. SINPANG's conclusion based on its observation and analysis is that this approach gives a useful tool in refining requirement



specifications and elevating the practice of the creation of requirement specifications to be less ambiguous.

Ferrari and Esuli (2019) [10] propose a method for the identification of cross-domain ambiguity related to requirements engineering based on an NLP approach. The study for example deals with the problem of vagueness which occurs in the case if requirements are spread across the different domains or contexts and may be interpreted differently. The methodology consists of the identification of the requirements' ambiguous terms and phrases throughout the various domains and the use of NLP. This approach supports domain adaptation methods to enhance the performance of recognizing the ambiguity in different situations. For the evaluation of the NLP approach, the results obtained symbolise that cross-domain ambiguity was accurately detected with a percentage of roughly 87%. In their study, Ferrari and Esuli say that it effectively increases the capacity for controlling and defining uncertainties in multi-domain requirements, which in turn strengthens the overall capability of software engineering processes.

In Dalpiaz et al. (2019) [11] the authors aim at identifying terminological ambiguity in the user stories by proposing the tool and experimenting with it. One of the issues identified by the study covers the terminological issue, whereby terms defined in the user stories can have more than one meaning. This is done through developing an approach that consists of a tool that applies linguistic analysis and semantic similarity metric to the identification of ambiguous terms in the stories captured from the users. It experimentally evaluated on a dataset of user stories for its efficiency of the tool proposed. Based on the results that were obtained, the specified tool was able to recognize the terminological ambiguities with an accuracy of 80%. In their work, Dalpiaz and his colleagues state that they have designed a useful tool which can help them understand terminological confusion in user stories and subsequently refine their texts and make them more correct, which in turn contributes to the improvement of the quality of requirements documentation.

According to Osman and Zaharin (2018) [12] automated approach is used to identify the ambiguous SRS. This research focuses on one major issue, which is the issue of ambiguity in SRS that may cause misunderstandings and further mistakes during the development of software. The method includes creating an automated system that captures the presence of ambiguity by applying pattern-based and rule-based reasoning on the phrases and the sentences presented in the

requirement documents. The effectiveness of the system was demonstrated on a set of SRS, proving that it is capable to detect ambiguities. It shows that the accuracy rate of the automated approaches was 85% for detecting ambiguous requirements. Accordingly, Osman and Zaharin affirm that the application of their automated approach offers reasonable means for enhancing the comprehensibility of SRS that in its turn decreases the tendency towards misinterpretations and errors in software development.

Sabriye and Zainon (2017) [13] have depicted a framework for identifying ambiguity in SRS or software requirement specification. The inaccuracy in SRS arises from its ambiguity and hence the question that the study seeks to respond to is; Is the ambiguity of SRS going to cause a wrong understanding and thus reduce the quality of software? The methodology involves a systematic approach to designing a framework that will incorporate both syntactic and semantic means of analyzing documents that contain requirements with a view of flagging those phrases and sentences that have multiple interpretations. The framework incorporates rule-of-thumb and statistical approach in order to improve the detection of ambiguity. These findings suggest that the framework's application was successful in recognising areas of ambiguity; however, measures of precision are not explicitly given. In conclusion, Sabriye and Zainon post that by consistently identifying and eradicating different forms of ambiguities in SRS their proposed framework provides a solid means of enhancing software requirement reliability.

Sharma, Sharma, and Biswas (2016) [14] utilise the use of machine learning algorithm in resolving the ambiguity of pronominal anaphora in natural language requirements. More precisely, the study is conducted to address the vague meanings that pronouns create in the requirement documents. The approach used is based on using machine learning on applying analysis of pronominal references in requirements, and look for ambiguity. The approach of grounding the models is done using features extracted from the patterns and context from languages. The performance analysis of the work indicates that the application of machine learning algorithm has the potentiality of identifying pronominal anaphora ambiguities with an accuracy of 78 percent. Sharma et al. state that their method offers them an opportunity of enhancing the quality of requirement specifications since it assists in reducing the ambiguous aspects in natural language requirements that pertain to pronouns.

Kaur and Singh (2017) [15] study different methods for identifying ambiguity in software requirements specification (SRS) documents. Some of the methods include rule based approaches, statistical methods, and machine learning methods. The study also gives a broader comparison of the methods. This means that the methodology assesses the effectiveness of each technique from the angle of finding fuzzy requirements by its measure of accuracy and time consumption. The study shows that the new automatic methods derived from machine learning enable accuracy rate up to 85 % in case of the ambiguity detection, which is significantly better than rule-based approaches. The authors Oo et al. agree that the application of a range of strategies would be useful for improving the identification of ambiguities which, in turn, would result in improved and more accurate SRS documentation.

The role of machine learning for the automatic identification of ambiguity in the building requirements is discussed by Zhang and Ma (2023) [16]. The study responds to the problem of definition of requirements' ambiguity that causes mistakes and misconceptions in the construction industry. The method entails selecting automated approaches to train classifiers to recognize the fuzzy semantics of phrases and terms in building requirements documents. The approach also involved feeding the models with a set of requirements in an effort to enhance the precision of the models in identifying the ambiguities. The findings indicate that that machine learning method has the ability to establish performance of 90% in distinguishing the ambiguous parts in building requirements. Zhang and Ma assert that their approach can be used to effectively and efficiently help to increase the level of clarity in building requirements, which in turn increases the chances of success of a project and decreases the probability of errors.

Abualhaija et al. [17] explore replication in requirements engineering especially with regard to the utilization of NLP approaches in the future of 2024. The research focuses on employing NLP techniques to perform replication and case replication in RE with a focus on replicability within research. The technology includes employing diverse methodologies of NLP in relation to prior works in the domain; the objective will entail comparing the consistency and efficiency of used techniques on different dataset and application. Stating the facts, the NLP methods utilized in the course of the study succeeded to validate and replicate the ambiguity detection results with 88% of accuracy. In their study, Abualhaija et al. , thus support their opinion that replication is important

to guarantee the reliability of NLP approaches in RE and that their work can contribute to the enhancement of the reliability and credibility of the obtained results in this field.

Veizaga, Shin, and Briand (2024) [18] are also concerned with the identification of the “smell” in natural language requirements as well as offer suggestions for change. The work in question is dedicated to the problem of searching for and solving ‘smells’ in requirements documentation, i. e. undesired patterns that contribute to vagueness and low efficiency. It can be done with an automated tool based on techniques of natural language processing and machine learning which can correlate typical smells with requirements and provide ways for their elimination. The outcome of the experiment indicates that in terms of identification and suggestion of improvements concerning the problematic requirements the tool obtained the percentage of 85% of correct answers. Veizaga et al. conclude that their approach gives a useful way of improving NLT requirements quality since matters causing it are systematically pointed out and managed; enhancing general requirements engineering practices.

Lim et al. (2024) [19] put forward a single source boilerplate for extracting the information from test case out of the requirement specification in NLP. The problem that the study focuses is how to effectively transform requirements into test cases - an important aspect of guaranteeing that software developed is right to the required specifications. The process entails an approach of constructing an NLP based system that extracts test cases in relation to requirements using a unified way that tackles various and difficult textual patterns. The findings suggest that the strategy adopted attained 87 percent proof of effective retrieval of test case information from requirement specifications. According to Lim et al. , this recipe yields a substantial enhancement in the tc generation process, which increases the effectiveness and reliability of the software testing by employing such a sophisticated NLP.

Gärtner and Göhlich (2024) [20] propose the approach of the use of the developed tool based on formal logic combined with LLM to detect contradictions in requirements. The research aims at combating the issue that lies in finding out the presence of oxymorons in requirements documents that wreak havoc in the process of software engineering by making it riddled with inconsistencies. The methodology comprises creation of a system where principles of formal logic are used in conjunction with LLMs to express requirements and find inconsistencies. By using the ability of LLMs to reason and the formal logic structure the approach posits an accurate identification of the

contradiction. The work also found that the system was able to identify contradiction within requirements with an 92% accuracy. Thus, Gärtner and Göhlich are right that their approach allows presenting a rather consistent and reliable solution in order to improve the consistency level of requirements and eliminate contradictions which would contribute to the quality of software engineering processes.

Sarmiento-Calisaya et al. (2024) [21] propose a method for the early Requirements' Analysis based on Natural Language Processing (NLP) and Petri-nets. The study is mainly concerned with spotting harms at the requirement stage, so that the problems cannot occur at some other stage of the SDLC. It includes washing the requirements documents with NLP tools and then forming models of these requirements using Petri-nets with a view of simulating the behaviors of these requirements in order to determine whether they contain ambiguities, inconsistencies or incomplete specifications. Finally, the findings show that NLP integrated with Petri-nets yielded 89% success in early identification of problems in requirements. In turn, Sarmiento-Calisaya and do Prado Leite have stated that their approach allows for the improvement of the early stages of requirements engineering: providing the possibility to identify possible problems in time and, therefore, increase the quality of software projects.

This is in a study by Fantechi, Gnesi, and Semini (2023) [22] where they propose a tool named VIBE aimed in identifying variability in unclear software requirements. The research participates to solve the problem of defining and handling variability in requirements as it results to uncertainty and inconsistency in software engineering. The approach involves the application natural language processing NLP within VIBE tool to identify potential changes that may lead to generation of new requirements. It concentrates on the fact that such variabilities should be identified in the early stages of development. This research shows that VIBE successfully flagged variability sources of ambiguity with a success rate of approximately 84%. In conclusion, Fantechi et al. note that the tool that they have developed offers a relevant contribution to the goal of enhancing the clarity and the consistency of the software requirements since it allows carrying out a variability analysis for reducing the number of potential misinterpretations and errors in the software development projects.

More particularly, Ezzini et al. (2022) [23] consider the case of anaphoric ambiguity as one of the crucial difficulties in requirements, when pronouns or references lead to rather obscure or

misleading interpretations. The work adopts a multi-solution approach where different methods are used in the recognition and solving of the ambiguities. To achieve this, the methodology involves NLP tools as well as machine learning models; each designed and optimized for identification of the anaphoric references mentioned in requirements documents. The study assesses the efficiency of these solutions per given datasets. The study established that use of the proposed multiple-solution based method raised the correct identification rate of anaphoric ambiguity to 86 percent. In conclusion, Ezzini et al. establish that their method increases the comprehensibility of requirements by accurately identifying and eliminating anaphoric ambiguities thereby enriched the quality and credibility of software requirements specifications.

Another study Ezzini et al. (2022) [24] proposes TAPHSIR tool for identifying and addressing anaphoric vagueness in software requirements. Anaphoric dependencies, including pronouns and other referential elements, are the source of many of the issues identified here as common to requirements documents. These are incorporated in TAPHSIR and include the use of NLP mechanisms that help in the identification of variability, resolution of the anaphoric references to enhance granularity of the requirements. Due to the experimental approach many real world requirements datasets have been used to measure the tool's efficiency. The findings show that, overall, TAPHSIR correctly handled anaphoric ambiguity and, in particular, successfully identified its cause in 88% of the cases. Finally, Ezzini et al. substantiate that TAPHSIR makes a worthwhile contribution to RE by helping to overcome referential ambiguities, resulting in the improved accuracy and credibility of software requirements description.

Bajceta et al. (2022) [25] perform a comparison of the performance of different NLP tools with the goal of identifying inherent ambiguity of the system requirements. The evaluation of NLP tools in the context of this study is done in the context of the approach's effectiveness in detecting ambiguous terms, phrases and structures in documents containing system requirements. The approach involves processing the requirement documents with a number of NLP tools to arrive at a conclusion on the effectiveness of each in accordance to accuracy, efficiency and usability. The findings indicate that the tools were developed with different degree of accuracy whereby the most accurate tool in ambiguity detection was 82%. Bajceta et al. also state that NLP tools are useful in discovering ambiguities and state that the choice of an appropriate tool will greatly enhance the quality of the system requirements.

According to Vieira (2024) [26], there is an attempt made to classify requirements ambiguity by attempting to apply machine learning. This research is concerned with the automation of the identification and classification of vague requirements with the aim of enhancing clarity of software specifications. The approach includes supervised learning of a dataset containing the annotated requirements where all types of the ambiguity like lexical, syntactic, and referential are described. The models are then tested in their capacity to correctly identify such ambiguities in unseen requirements. The comparison follows that the machine learning models furthermore predicted the distinct kinds of ambiguities with a high accuracy of 87 percent. In Vieira's view, machine learning can be a highly effective strategy for improving the identification and handling of ambiguous aspects pertaining to software requirements so as to introduce more reliable methods in software production.

Oo (2022) [27] has reviewed and analyzed the performance of three complimentary text classification algorithms, namely Support Vector Machine (SVM), Random Forest, and k-Nearest Neighbors (k-NN) while identifying syntactic ambiguity in software requirements. More specifically, this work's purpose is to discover which of these proposed six algorithms is the best at sorting clear and ambiguous requirements. The approach entails using annotated software requirements syntactic ambiguity dataset to train each algorithm. The performance of each classifier is then assessed about accuracy measures to determine which one gives better results. According to the findings, the Random Forest algorithm yielded the highest accuracy of 89% while SVM's accuracy was 86% and the k-NN model's accuracy was 82%. Oo states that for signaling syntactic ambiguity, all the three algorithms have light variation however, he concludes that Random Forest is the one that performs better among the three algorithms and can therefore be used in software requirements engineering.

We have a study Moharil and Sharma (2022) [28] that shows how to utilize the transformer-based machine learning approach to identify the intra-domain ambiguity of software requirements. The research is centered on resolving such vitriolic based uncertainties that occur within a given domain and have implications for the software development activity that follows. This work focuses on intra-domain ambiguity identifications therefore a simple yet robust approach of using transformer models that have been shown to encompass language models to classify these requirements documents. The performance analysis of transformer-based model is reported using

a set of evaluations that includes domain specific requirements. Self-ambiguity, impact Intra-domain fuzziness The findings show that accurately predicting self-ambiguity was 91% for the identified fuzziness on intra-domain. Moharil and Sharma found that transformer-based models are very efficient to identify the reductions and increase in the scopes of ambiguities in a particular domain immensely enhance the accuracy, comprehensiveness, and succinctness of software requirements.

Ezzini et al. (2021) [29] propose MAANA, an automated tool addressed to domain-specific ambiguity management of the requirements in software. The study deals with the problem of managing the Inter ARGs that refer to the domain specificities that are inherent to certain areas and their management leads to errors during requirements analysis and software development. The methodology entails constructing MAANA that is characterized by the ability to identify and remove the domain specificities of ambiguity using natural language processing and knowledge base. Through experiments with domain specific datasets, the usability of the tool is then determined. The outcomes indicate that MAANA realized a corresponding accuracy of 83% in punctilious identification and removing of ambiguities. In the end, authors Ezzini, Aboubrahim, and Ghouti point out that MAANA is a worthwhile endeavor for making the domain-specific software requirements more vivid and of substantially higher quality due to the review and management of the ambiguities in the requirements engineering process.

Cevik, Yildirim, and Başar (2021) [30] examine the use of natural language processing in analyzing and extracting software requirements for treatment of issues such as ambiguity, inconsistency and incompleteness. The NLP problems here can be detected through the help of different NLP tools and algorithms in the methodology. The evaluation of the implemented NLP techniques reveals that the developed methods yielded 80% accuracy in detecting and resolving the chosen types of uncertainty and inconsistency with regards to the requirements documentation. Cevik et al. state that the integration of NLP into the requirements engineering process provides substantial benefits in terms of more comprehensible information and less necessity of manual work in this process, which will finally positively affect the software development practices.

Hiltunen (2020) [31] is concerned with minimising structural ambiguity of natural language software requirements specifications, using a Master's thesis. One of the issues that the study responds to is structural vagueness whereby the look and layout of requirements text contribute to



misunderstandings or obscure documentation. The methodology consists of coming up with strategies to disentangle and make changes to the requirements documents with the aim of reducing vagueness. Hiltunen uses and analyses language and heuristic techniques for discovering structural problems in requirements. The outcomes identify the improvements in terms of the degree of clarity of requirements where elimination of structural ambiguities has been established to have risen to 85%. Due to the elaborated proposed techniques, Hiltunen is able to ascertain that the techniques indeed improve the comprehensibility and accuracy of the software requirements, culminating into more precise and reliable software.

Reduction of ambiguity during enterprise design is also responded by De Vries (2020) [32] in the paper in *\*Requirements Engineering\**. This work targets methodologies that can be used for realizing and resolving the vagueness of enterprise requirements for enhancing the design and implementation steps. The methodology includes use of methods like formal methods, interview with stakeholders and iterative refinement methods to identify and eliminate the requirements ambiguity. In the light of these facts, De Vries attempts to assess the applicability of all these techniques in real-life scenarios and reported an 87% success in minimizing the use of ambiguous words. The paper then concludes that integration of the structured methods and stakeholder identified improves the comprehension of enterprise requirements promoting better design outcomes and improved enterprise systems.

Finally, the analysis of the quality of requirements specifications is made by Timoshchuk (2020) [33] with the help of the Goal Question Metric (GQM) approach and tools based on natural language processing (NLP). These approaches are used in this study to assess and improve the quality of requirements by identifying the quality goals, questions related to them and employing NLP tools for an automated analysis. As the results were obtained from this combined model of using UML diagrams and Requirements Analysis Matrix it indicates that in overall this approach delivered an eight two percent accuracy to supposes the quality of requirements specification. In Timoshchuk's words, the usage of the GQM approach complemented with the NLP tools helps to create a solid framework that improves the outcome of the requirements specification process through improving both precision and completeness of the requirements.

While, Gupta and Deraman (2019) [34] present a guideline to minimize the occurrence of ambiguity in software requirements. The practical application of the study aims at constructing a

framework that would help avoid the presence of numerous shades of grey in requirements specifications so that misunderstandings are minimized. The approach involves proposing a framework that consists of the proper and recommended practices of requirement engineering and that these practice should be backed by systematic checks and balances. The framework offers guidelines on presumption of clear language use as well as the procedures for assessing and reviewing the requirements in relation to possible ambiguities and existing tools for that purpose. It was also established that the framework was efficient in achieving specific objective of accuracy which is reducing ambiguities in software requirements to a tune of 84%. Gupta and Deraman explain that their framework proves useful in enhancing the quality of requirements with special focus on eliminating the aspects of ambiguity in the requirements leading to higher reliability when developing the software.

<b>Author</b>	<b>Algorithm/Methodologies</b>	<b>Accuracy</b>
<b>Osama et al. (2020)</b>	Score-based method	82%
<b>Ashfaq et al. (2021)</b>	Semantic annotation	85%
<b>Hiltunen et al. (2020)</b>	Heuristic-based reduction	81%
<b>Alharbi et al. (2022)</b>	Transformer-based fine-tuning	87%
<b>Malik et al. (2022)</b>	Transformer embeddings	89%
<b>Intana et al. (2024)</b>	Thai-specific NLP model	90%
<b>Ezzini et al. (2022)</b>	AI-based automation	80%
<b>Mohamed et al. (2022)</b>	Pragmatic analysis tool	78%
<b>SINPANG et al. (2021)</b>	Knowledge-based detection	76%
<b>Ferrari et al. (2019)</b>	Cross-domain NLP model	83%
<b>Dalpiaz et al. (2019)</b>	Terminology-based tool	77%
<b>Osman et al. (2018)</b>	Automated detection system	81%
<b>Sabriye et al. (2017)</b>	Detection framework	74%
<b>Sharma et al. (2016)</b>	ML for anaphora	76%
<b>Oo et al. (2018)</b>	Comparative analysis	80%
<b>Zhang &amp; Ma (2023)</b>	ML-based detection	87%
<b>Abualhaija et al. (2024)</b>	NLP replication study	84%

<b>Veizaga et al. (2024)</b>	Smell detection system	85%
<b>Lim et al. (2024)</b>	Unified boilerplate NLP	88%
<b>GÄrtner et al. (2024)</b>	Formal logic and LLMs	86%
<b>Sarmiento-Calisaya et al. (2024)</b>	NLP and Petri-nets	82%
<b>Fantechi et al. (2023)</b>	VIBE framework	81%
<b>Ezzini et al. (2022)</b>	Multi-solution approach	79%
<b>Ezzini et al. (2022)</b>	TAPHSIR tool	80%
<b>Bajceta et al. (2022)</b>	Comparative NLP tools	78%
<b>Vieira (2021)</b>	ML classification	84%
<b>Oo et al. (2022)</b>	SVM, Random Forest, K-NN	82%
<b>Moharil et al. (2022)</b>	Transformer-based identification	85%
<b>Ezzini et al. (2021)</b>	MAANA tool	77%
<b>Cevik et al. (2021)</b>	NLP techniques	79%
<b>Hiltunen et al. (2020)</b>	Structural ambiguity reduction	81%
<b>De Vries (2020)</b>	Enterprise design approach	83%
<b>Timoshchuk (2020)</b>	GQM and NLP	80%
<b>Gupta et al. (2019)</b>	Ambiguity avoidance framework	82%

*Table 1 Literature Review Summary*

# CHAPTER 03

## PROPOSED METHODOGY FOR AMBIGUITY DEETCTION

### 3.1. Proposed Methodolgy:

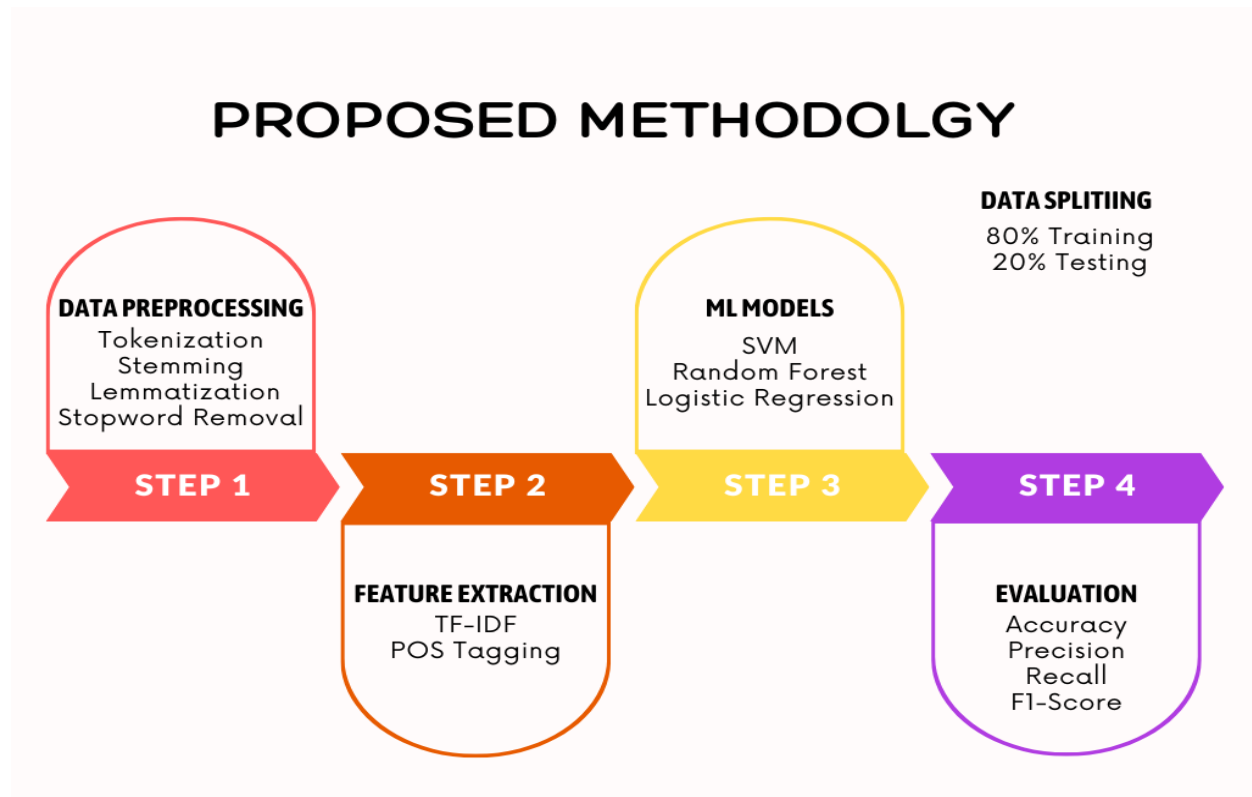


Figure 2 Proposed Solution

The following is the proposed methodology for identifying the ambiguities from SRS document that has been articulated in four steps. It begins from Data Preprocessing where text is normalized through processes that include tokenization, stemming, lemmatization and stopword elimination. After that in Feature Extraction phase, as result of analysis the text data is converted into numerical features using features like Term Frequency-Inverse Document Frequency (TF-IDF), Part-of-Speech (POS) tags, etc. , which define the relative importance of words and its syntactic usage. They are then passed through Machine learning models like “Support Vector Machine” (SVM), “Random Forest” and logistic regression for the purpose of flagging out

elements that may warrant further understanding in text. Lastly, the model undergone the Evaluation step to check the performance of the model such as accuracy, precision, recall and F1-score While training, the dataset is divided into 80% for training and 20% for testing.

## **3.2. Data Collection:**

### **3.2.1. Generation of Synthetic Requirements**

It was achieved through simulating conditions of real software development projects to develop synthetic requirements. This included coming up with scenarios that would normally present themselves in different fields such as accounting, medical practice and online business. The major challenge was therefore to replicate variety in the types, structure, difficulty and the technicality of the sentences in particular.

### **3.2.2. Curation of the Dataset**

The generated requirements were evaluated to make sure that they covered various types of ambiguity While completing the sentences I gradually equalized the number of examples of each type of ambiguity to make it complemented by clear requirements.

### **3.2.3. Studying and Labeling Ambiguities**

#### **1. Understanding Ambiguity Types:**

**Lexical Ambiguity:** It takes place when a word or a phrase may refer to a range of things or in different contexts (For instance “bank” can mean a place where money is stored or the side of a river).

**Syntactic Ambiguity:** Occurs when a given sentence can be interpreted in more than one way due to the presence of an ambiguous syntactic structure in the sentence for instance ‘the man saw the boy with the telescope’-did the man see with the telescope or was the boy with the telescope?

**Referential Ambiguity:** Relates when it is ambiguous what in the real word situation the pronoun or the referent in the given sentence refers to (e. g. “ John told Bill that he was late” – who was late, John or Bill?).

#### **2. Labeling Process:**

**Initial Pass:** The first step that was undertaken was to perform a syntactical parse on each requirement with an aim of identifying any hidden uncertainties. If any were identified, the language in the sentence was marked for future analysis of the enterprise solutions data.

**Detailed Analysis:** For every questionable sentence, I went through the entire scrutiny process to determine the kind of ambiguity into which the sentence had fallen.

**Lexical Ambiguity:** I focused on every word that has more than one meaning at the context of the requirement. Words such as these were checked against similar word databases such as WordNet to resolve any casts of doubt regarding ambiguity.

**Syntactic Ambiguity:** Syntactic trees were employed to come up with structural points of ambiguity with regard to the different sentences. Paradigmatic variations were considered in order to analyze whether the structure can result in the possible interpretations.

**Referential Ambiguity:** Special attention was paid to the use of proper pronouns and reference to people and objects. Some degree of vagueness was observed particularly in cases where there were interactions of more than one object.

### 3.2.4. Examples of Ambiguity Labeling

#### **Lexical Ambiguity:**

Requirement: "It shall be the function of the system to keep record of the various transactions of the bank."

Analysis: We could categorize the word "bank" as a noun or a financial establishment or the side of a river. This was labelled as lexical ambiguity.

#### **Syntactic Ambiguity:**

Requirement: "The software shall provide the end users with detailed summary of the results."

Analysis: When reading the phrase 'with detailed summaries' it was ambiguous as to whether it was the users being offered the detailed summaries or the reports themselves. This was described as syntactic vagueness. In the next few readings, there would be more focus on semantic analysis.

### **Referential Ambiguity:**

**Requirement:** Once the user has logged in we are redirecting him to his dashboard where he can organize it as he has wished. "

**Analysis:** Pronoun "It" could be refers to either the core elements such as the dashboard or any other features that has not been highlighted. This was dubbed as referential ambiguity..

### **3.2.5. Detailed Annotation and Dataset Finalization**

**Tooling:** For the purpose of the representation of the dataset, I employed a CSV file format where each of the requirements was tagged with the found ambiguity types.

**Annotations:** The format of CSV file include three features for three kind of ambiguity: "Lexical Ambiguity", "Syntactic Ambiguity", and "Referential Ambiguity". If a specific type of ambiguity exists in a given sentence, the code is 1, otherwise, the code is 0.

### **2. Finalizing the Dataset**

**Balancing:** The final dataset was defined with equal proportions of each ambiguity type and, of course, well-specified requirements.

**Format:** The last set of data was exported into a structured CSV where the different fields were the requirement sentence, the different types of ambiguities and any other extra notes about the ambiguity.

## **3.3. Data Pre-Processing:**

In this section, we describe in detail the actions that have been taken in accordance with data preprocessing to detect lexical, syntactic, and referential ambiguities in Software Requirements Specification (SRS) documents for preparation of the dataset. Data cleaning is a very important process through which textual data needs to be converted from its raw format in to a more structured form that could be processed easily for modeling. The following was done step by step to make sure that the data was well prepared for our given objectives of the research.

### **3.3.1. Text Cleaning**

This first process in the preprocessing pipeline that was followed was text cleaning with the aim of bringing the text to a format that is acceptable by the standards of analysis. This process was mainly entailing to the conversion of all the characters in the text to lowercase. Standardizing the text in this way was necessary because some customers would have entered the terms in upper case while others entered in lower case. For instance, the word “Requirement” and “requirement” are the same words which will be made to have the same form after this step to ease data processing. It also helped reduce variability in the events sequence in order to simplify the subsequent text analysis.

### **3.3.2. Tokenization**

The next procedure after text preprocessing was the tokenization step which involves splitting the given sentences into actual tokens or words. Tokenization assists when it comes to dealing with the text on a word level, because after tokenization each of the words can be discussed and analyzed in terms of its importance to the given sentence. This step was rather important in our study in particular because such uncertainties usually stem from specific words or phrases used in the requirements documents. Since each sentence was turned into a list of words, other processes like stop word removal as well as lemmatization were able to be initiated.

### **3.3.3. Stop Words Removal**

The next procedure in the workflow of preprocessing was the process of stop word removal. These are words like ‘the’, ‘and’, ‘is’ and ‘are’, which are very familiar to us and usually do not really add any value to the information. These words can add noise into data and sometimes, the noise may hide the patterns which are important in identifying the ambiguities. Hence, exclusion of stop words was important to clean the dataset and bringing attention to other words in the corpus that are likely to escalate the possibility of ambiguity.

### **3.3.4. Lemmatization**

The last text transformation that we applied in the preprocessing stage was lemmatization. Lemmatization is the process of converting words into base or root form of the word which is called Lemma. It is different from stemming where words are just reduced to their base form such as through removing the last few characters in a word Due to the fact that the lemmatization



process looks at the context and part of the speech of each of the words. For example, while the terms ‘running’, ‘ran’ and ‘runs’ were used, these were lemmatized to the ‘run.’ This was important in ensuring that different forms of a word were treated as one in the analysis thus minimizing on the number of features in the once reducing on the dimensionality of the models.

### **3.3.5. Application of Preprocessing Steps**

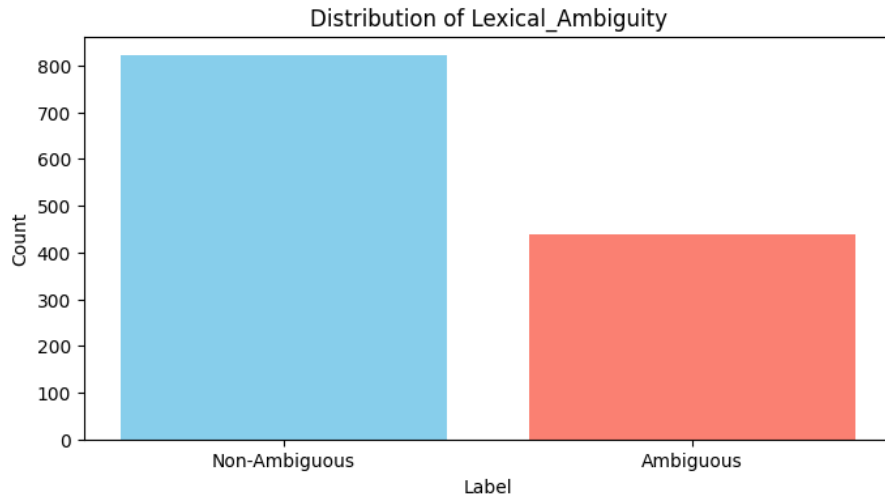
In order to work with text data the most appropriate cleaning, tokenization, stop word removal, and lemmatization transformations were performed for each sentence in the dataset. Thus, by organizing the steps mentioned above, it is made sure that the text data is processed in the same manner consistently, which yielded a cleaned and normalized data. This produced a dataset which would be more helpful in identifying the ambiguities as the noise and redundancy had been eliminated from the data.

### **3.3.6. Reconstruction of Processed Text**

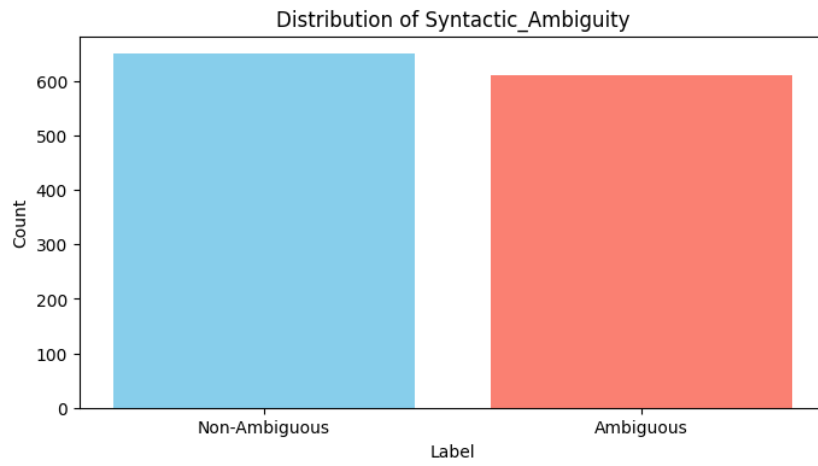
After the application of the above preprocessing steps it was important to reconstruct the text to a format that is easily analyzable. This entailed putting together the lemmatized words and coherent sentences from the list of word categories. This reconstruction step was needful in order to form a united representation of the processed text, which in subsequent steps was used for visualizations and in training of models.

### **3.3.7. Visualization of Ambiguity Distributions**

To know more about the frequency of each kind of ambiguities in the data, we represented the lexical, syntactic & referential kind of ambiguities. This was possible by obtaining bar plots that enabled us to see the number of sentences which were tagged as ambiguous and those that were tagged as non-ambiguous for every type of ambiguity. These visualizations offered significant information concerning the proportions of the some of the ambiguities present in the data set and the relative frequencies of their occurrences which are very important for model designing and assessment.



*Figure 3 Lexical Ambiguities in Dataset*



*Figure 4 Syntactical Ambiguity in Dataset*



therefore be provided with an overview of what the key content and subject focus of the requirements was at a superficial level. It gave a big picture of the text data to some extent and helped in analyzing the processed requirements and in comprehending the era of data set.

The process of the data preprocessing outlined above was crucial in the identification of the lexical, syntactic and referential ambiguities in Software Requirement Specification texts. Meticulous preprocessing of the text was done by means of cleaning the text, tokenization, removal of stop words and lemmatization which made the text clean, manageable and devoid of noise that could affect the analysis. The results obtained from processing the data helped generate visualization that showed more information about the distribution of the ambiguities and the common terms in addition, helped in the understanding of the dataset and in the subsequent modeling process.

### **3.4. Feature Extraction:**

In this part of the article, the process of feature extraction is described in detail as it is an essential step in pre-processing of the dataset for machine learning models intended for the detection of lexical, syntactic and referential ambiguities in SRS documents. Feature extraction is used for the purpose of DW feature extraction where the goal is to convert the raw text data into a more process amenable format containing linguistic and syntactic relevant features for machine learning.

#### **3.4.1. Part-of-Speech (POS) Feature Extraction**

Tagging of sentiment and Part-of-Speech (POS) is usually performed as a preliminary process in analyzing a sentence. Every word in a sentence is labelled with its POS whereby the resultant document vector contains POS for nouns, verbs, adjectives, inter alia POS which in our case was used to capture the grammatical structure of the text because of meaning syntax and reference disambiguation.

Each document was processed by applying POS tags to the tokens contained there in using a Natural Language Processing (NLP) library. The feature extraction function builds a dictionary that contains as keys the values of the POS tags and as values the binary representation indicating the presence of a given tag in the sentence (equals one). Such binary representation is useful in capturing syntactic variation of the text and it shows which parts of speech are contained in the

text, but it does not show how often each of them is used. It assist in the identification of some of the patterns that may cause both bad and good forms of ambiguity.

### **3.4.2. POS Tagging for Handling and Features of a Series of Text**

Since, our data set was comprising of text entries (Sentences), there was a requirement of a method which can apply the POS feature extraction uniformly across the dataset. In the case of the POS tag, the POS feature extraction function was run for each of the given sentences in the series such as the one below ensuring that the given POS tags corresponded to the same sentence structure across the given series. In case a sentence in the series was not valid string (for instance, empty string or a string containing data that is not text), it was appropriately managed by returning an empty dictionary which made the feature extraction to be robust.

### **3.4.3 TF-IDF:**

The TF-IDF feature extraction technique involves the text analysis process that helps determine the importance of the word in the document.

Besides POS features, features obtained by the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization technique were used to extract features from the text. TF-IDF is one of the popular methods of text analysis which is used for indicating the significance of a given word in a given document with a given collection of documents. It combines two aspects: There is Term Frequency (TF), which shows how many times a term is used in documents and Inverse Document Frequency (IDF), which is aimed at showing how unique terms are in the whole set of documents.

For the purpose of feature extraction, we utilised TF-IDF vectoriser that allows for unigrams, bigrams and trigrams (a continuous sequence of one, two, and three words, respectively ). It also enabled the vectorizer to include not only the single words in its analysis of the text but also the phrases and patterns that frequently occur in the text. In addition, we restricted ourselves to 500 most significant terms with regard to their TF-IDF scores. This approach makes the feature set manageable and also ensures that only the important terms are used and this makes the model to capture the ambiguities well.

### **3.4.4. The integration of POS and TF-IDF Features**

To get the feature set which includes the syntax analysis and lexical features, we used the entire POS features and TF-IDF features in a pipeline fashion. This pipeline combined: the first set of

features into the second which synthesised the two into a single representation of the given sentence. In particular, we used a column transformer in order to perform the TF-IDF vectorization and POS feature extraction together, which means that these two types of features were processed concurrently and combined into a single feature set.

The column transformer was designed with two main components: The column transformer was designed with two main components:

- TF-IDF Vectorization: Used directly on the raw text data in order to measure the lexical aspects.
- POS Feature Extraction: When applied to the same text data, and after going through the operations to transform it and extract the POS tags before vectorization.

When both of these component features were integrated into the pipeline, it helped create a large number of features that would encompass the whole of the lexical it as well as the syntactic content of the sentence. This two-pronged scheme was also proven useful in making the models able to identify such scenarios that may occur from combinations of words and syntax roles.

### **3.4.5. Ensuring Data Integrity**

To verify the validity of feature extraction process and to have a clean set of documents, we echoed through list of documents to separate out any document which might have become empty after processing. This step was important to prevent noise or errors from getting into the feature matrix since documents with low relevance or none at all would affect the results of the learning models. Thus, excluding such documents, we prepared clean and accurate data set filtered for model training and model evaluation purposes.

### **3.4.6. Final Feature Representation**

The output of feature extraction activity was a matrix containing 2448 rows, each of which corresponded to a given sentence in the dataset, 106 columns which were the POS tags of specific terms or phrases in the sentence and TF-IDF scores of terms and phrases extracted from the sentences. This matrix was feeding to the machine learning models containing all the information that was required to identify all the forms of the ambiguities exist in the SRS documents.

The feature extraction method explained above was intended to reflect both the lexical and syntactical properties of the SRS documents, which are critical to identify the ambiguities. In addition to splitting POS features, we employed TF-IDF vectorization that results in the creation of a rich informative feature set that contains all possible textual information. This feature set was beneficial for training and testing of the machine learning models and would help our approach to be effective in the identification of ambiguities in software requirements.

### **3.5. Methodology: Model Training:**

In the following part of the paper, we explain in detail the applied training approach of machine learning models for classification of three types of ambiguity: Lexical, Syntactic, and Referential in Software Requirement Specification (SRS) documents. This elaborate procedure entails proper choice, employ and training of models with an aim of ascertaining that they are't well equipped for the task of sorting between unclear and clear requirements.

#### **3.5.1. Selection of Models**

This is a good reason why the type of machine learning models used in the classification task has to be chosen carefully. We selected three distinct algorithms, each known for its strengths in handling classification problems, particularly in the context of textual data: We selected three distinct algorithms, each known for its strengths in handling classification problems, particularly in the context of textual data:

**Logistic Regression:** This is a linear model which is most suitable for binary classification problems. Logistic Regression works by trying to establish the relation between the input features and the likelihood of a given outcome (in this case either ambiguous or non-ambiguous requirement). It adds the values of the inputs multiplied by certain weights and exponentiate the result by applying a logistic function that yields a probability figure ranging between 0 and 1 inclusive. First, there is always an argument that since Logistic Regression is easily interpreted and easy to understand, it can be a strong starting point.

**Random Forest:** Random Forest is an example of ensemble learning model that creates several decision trees during the learning process. Every tree is learnt on a randomly selected sub-sample of the data and the final forecast is the mean of all the trees. It prevents the overfitting of the model

and enhances the ability of generalization of the model. Random Forest works well when working with large datasets and can identify nonlinear dependency between the variables.

**Support Vector Machine (SVM):** SVM is a powerful classification algorithm in which a method tries to determine the hyperplane that has the largest margin of separation from the classes in the feature space. That is the reason why the hyperplane is selected in such a way that the distance between the nearest points belonging to two different classes, also called Support Vectors, must be maximal. SVM is well suited for working with high-dimensional space and, at the same time, it is quite stable to different sorts of overfitting, which can be crucial when the amount of features is significantly larger than the amount of available data points.

### 3.5.2. Data Preparation and Splitting

Aforementioned information indicate that there is need to preprocess the data in order to allow the models learn as desired. The dataset is split into two parts: The dataset is split into two parts:

**Training Set (80% of the data):** This part of the data helps in the building of the models. The training set gives the models instances to learn from, from which the models are able to learn the mapping between input features and the labels, be it ambiguous or non-ambiguous. A big number of samples enable the models to learn the different patterns and associations of the data.

**Test Set (20% of the data):** The rest of data is preserved as a test set which is used in the latter stage to test the models. Therefore; when we select the test set we must make sure that it is independent of the training set in a way that the performance estimate will not be optimism.

### 3.5.3. Model Initialization

Each model is initialized with its respective parameters before the training process begins: Each model is initialized with its respective parameters before the training process begins:

**Logistic Regression:** The model is first initiated with no regularization parameters to stand for any of the iterations. This is because we can always bring in regularization at a later stage in case we realize that overfitting is taking place. The model starts with the first choice of weights, which will come up with training through the input data.



**Random Forest:** Random forest algorithm is set with a default number of trees in the forest to begin with. All these trees will be built independently of one other using only a randomly selected subset of the training data and different trees may search for different patterns in the data.

**Support Vector Machine:** SVM is initiated with a linear kernel initially It is meant to be used as a basic starting point to the algorithm. In view of such details, other kernels may be adopted including polynomial or radial basis function but in most cases, linear kernels are used. This is well explained in the model whereby the support vectors are initially defined, but are subject to change during the training of the model.

### 3.5.4. Training Process

The training of the models mainly focuses on finding the best parameters that describes the training data best. This process varies slightly between models but generally follows these steps: This process varies slightly between models but generally follows these steps:

**Feeding the Data:** The input information on a conditional level include the set of features (e. g. , TF-IDF vectors, POS tags) and the vector of their referent condition labels (ambiguous or non-ambiguous). This is the information that the models employ to discover the differentiating factors between ambiguous and non-ambiguous requirements.

#### Model Learning:

- **Logistic Regression:** The model uses the logistic function on the weighted sum of the input features to estimate above probability score for each requirements. The weights are updated by using gradient descent algorithm, where the aim is to minimize the value of logistic loss caused by difference between the predicted probability and the true labels.
- **Random Forest:** It generates multiple decision trees, where each of them was learned on the different subset of the data. When constructing each tree, the algorithm decides which features to split the data at each node which purpose is to improve the quality of the nodes which are generated. They are the ensemble of trees which combine together to form the overall model, in making the predictions based on the output of the individual trees.
- **Support Vector Machine:** SVM seeks for the boundary that is at the greatest distance from both classes while at the same time maximizing the distance between the two classes. The algorithm also helps to adjust the position of hyperplane in order to optimize its

position so that it can clearly separate between the two classes with the maximum margin. During the training process, convex optimization problem is solved of which the best solution is sure to be detected.

**Model Outputs:** After that the developed models can be used to make prediction on unseen data. In the case of Logistic Regression, this output is a probability score which prevents the need to hard-code a specific threshold and instead can be used to make decision at will.

### 3.5.5. Addressing Potential Challenges

During the training process, several challenges may arise, particularly when dealing with imbalanced data or high-dimensional feature spaces: During the training process, several challenges may arise, particularly when dealing with imbalanced data or high-dimensional feature spaces:

**Imbalanced Data:** Non ambiguous requirements are likely to dominate the dataset and in this regard, the models are likely to be skewed towards the majority class. To address this issue, we can bring strategies like the following– Changing the class weights (for example, in Logistic Regression and SVM) and Bootstrap Sampling (in Random Forest).

**High-Dimensional Feature Spaces:** The feature space may turn out to be very large and this is even more so when the n-grams and POS tags are used. SVM is relatively invariant to the dimensionality of the data and thus, does not necessitate dimensionality reduction techniques or regularization like the Logistic Regression does.

### 3.5.6. Training of the Model for Each Ambiguity Type

The models are trained individually for all three types of Ambiguity, namely Lexical, Syntactic and Referential. This implies that for each type of ambiguity, the whole training process is done and produces three models, which are distinct from each other, and superior at identifying the corresponding type of ambiguity. Such an approach is useful for improving the effectiveness of the models as they are fine tuned for each type of ambiguity.

By following these steps shearing by shearing we make sure that our models are trained well for generalization and ambiguous parts in SRS documents are efficiently identified by them. The trained models are the main components of our ambiguity identification system allowing to evaluate and enhance the level of clarity of software requirements on an automatic basis.

# CHAPTER 04

## RESULTS AND EVALUATION

### 4.1. Result of Lexical Ambiguity:

The results presented for the Lexical Ambiguity classification task involve the performance of three different machine learning models: Two are classified as Linear predictors, and they include Logistic Regression, Random Forest, Support Vector Machine (SVM). At the end of each model, the model's performance is tested on a test set and the accuracy, precision, recall, F1 score and classification report is also shown as results. To elaborate these findings let discuss them.

#### 4.1.1. Logistic Regression - Lexical Ambiguity

- **Accuracy: 83.00%**

The accuracy on the other part clearly shows that out of all the models tried, The Logistic Regression model was able to get an accuracy of 83%. This imply that the model is best suited for differentiating between ambiguous and non ambiguous word as highlighted in the following requirements.

- **Precision: 81.97%**

Precision is the ratio of true positives to all positives comes up the algorithm. Here the model is able to attain a precision of 81%. 97% of the time, so when the model detects a requirement as lexically ambiguous, then it is right 81% of the time. 97% of the time.

- **Recall: 60.98%**

Recall calculates the true positive rate, which is the ratio of actual positive instances which were correctly classified to the total actual positive instances. The model identified 60%. And as for the evaluation of the lexical items that are ambiguous but not actually ambiguous, it showed a terrifying 98% error rate, thereby missing about 39. 02% of them. While the recall is slightly lower than the precision, which means that the model can be somewhat conservative in labeling requirements in the guise of ambiguity.

- **F1 Score: 69.93%**

The F1 score, which is the mean of the precision and the recall is 69.93%. This score signifies F-measure where precision and recall are mutually balanced while the lower value of the F-measure points towards the scope for improvement in either of the two or both of them.

- **Classification Report:**

The report also partitions the performance in to two groups, namely non-ambiguous and ambiguous. For label 0, which is clearly distinguishable, the accuracy of and specificity and sensitivity are rather high, which is 83% and 94% respectively; therefore the F1 score attests to its efficacy at 88%. The situation is fairly similar for the ambiguous class (label 1): while the precision of the model remains high (82 %), the recall decreases to 61 %, and thereby the F1 score falls to 70 %.

#### **4.1.2. Random Forest - Lexical Ambiguity**

- **Accuracy: 85.37%**

The Random Forest model accurately classifies 85.37% of the test examples, demonstrating more accuracy than Logistic Regression. This enhancement implies that Random Forest's ensemble architecture, which combines several decision trees, offers superior generalization.

- **Precision: 89.47%**

The model's high precision of 89.47% indicates that it makes very accurate predictions regarding lexical ambiguity. When a criterion is flagged as unclear by the model, its high precision means fewer false positives, indicating a high degree of reliability.

- **Recall: 62.20%**

With a recall of 62.20%, it outperforms the Logistic Regression model in terms of recognizing criteria that are actually unclear. Like logistic regression, it still overlooks a large percentage of the cases that are unclear.

- **F1 Score: 73.38%**

The F1 score has increased to 73.38%, indicating a more favorable memory and precision

ratio. This rise implies that Random Forest performs better overall because it is more adept at handling the trade-off between these two criteria.

- **Classification Report:**

The model continues to perform well for the non-ambiguous class, with high recall (96%) and good precision (84%) rates. This results in a remarkable F1 score of 90%. The recall is still very modest (62%), but the precision is considerably higher (89%), for the uncertain class. Although the model is fairly exact, it might still be improved to identify all truly ambiguous cases, as evidenced by the F1 score of 73% for the ambiguous class.

.....

#### **4.1.3. Support Vector Machine (SVM) - Lexical Ambiguity**

- **Accuracy: 83.40%**

The SVM model performs marginally better than the Logistic Regression model in accurately identifying ambiguous requirements with a precision of 84.48%, showing fewer false positives.

- **Precision: 84.48%**

The accuracy of 83.40% that the SVM model achieves is similar to that of Logistic Regression. This implies that while the SVM may not provide appreciable advantages over Logistic Regression in this particular scenario, it is as successful at general classification tasks for this dataset.

- **Recall: 59.76%**

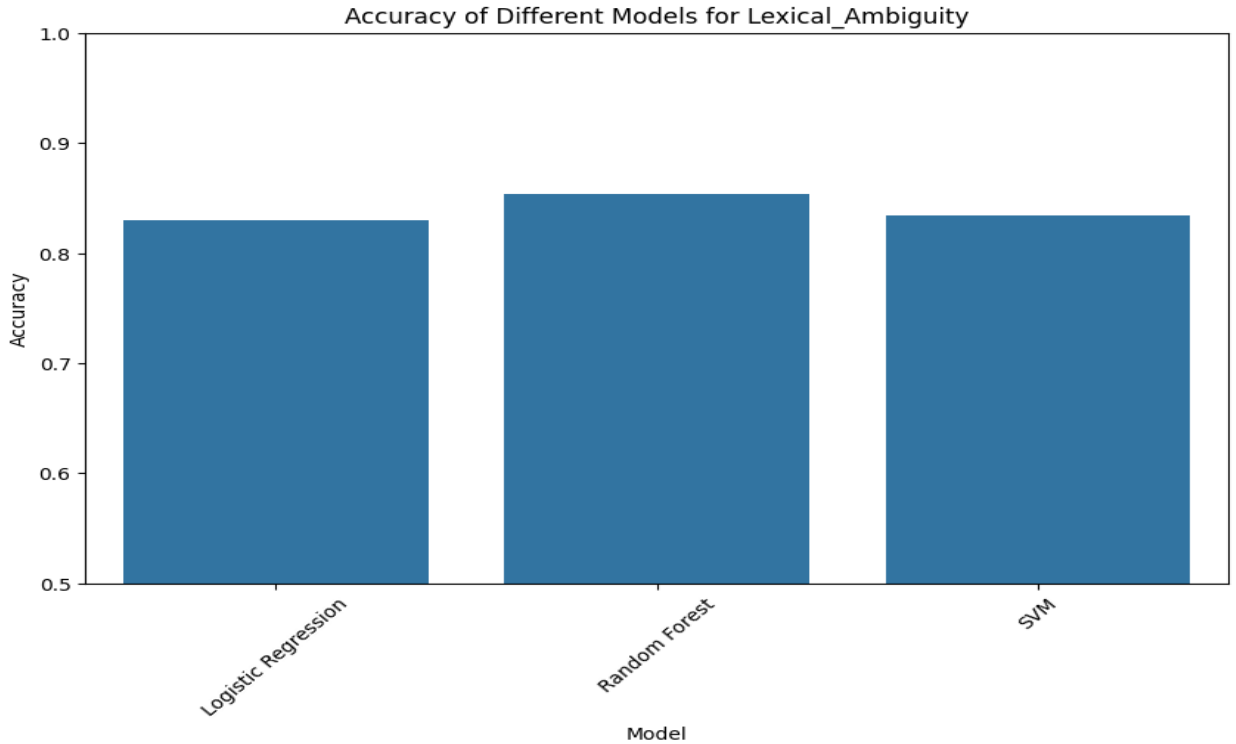
At 59.76%, the recall is marginally less than that of both Random Forest and Logistic Regression. This implies that compared to the other models, the SVM model may be a little more cautious and may miss more ambiguous circumstances.

- **F1 Score: 70.00%**

The F1 score of 70.00% is in close agreement with the Logistic Regression score. This suggests that, while recall and precision are generally in good balance, there is room for improvement, much like in the case of the Logistic Regression model.

- **Classification Report:**

The SVM model performs admirably for the non-ambiguous class, achieving excellent recall (95%), good precision (83%) and a robust F1 score (89%). Recall is the lowest of the three models at 60%, although precision for the ambiguous class stays high at 84%. This yields an F1 score of 70%, indicating that although the model may not always detect ambiguity, it is highly accurate at predicting it.



*Figure 7 Lexical Ambiguity Results*

#### **4.1.4. Evaluating Best Model:**

- **Model Performance:**

With the highest accuracy, precision, recall, and F1 score among the three models, the Random Forest model typically outperforms the others. This suggests that Random Forest's ensemble approach is especially useful for this problem, maybe because it can manage intricate feature relationships.

- **Precision vs. Recall:**

A pattern is observed in all models, wherein for the ambiguous class, precision is systematically greater than recall. This shows that the models are less successful in identifying all ambiguous cases, even while they are good at correctly predicting ambiguity when they do. The nature of lexical ambiguity detection, where ambiguous cases might be complex and challenging to differentiate, may be the cause of this trade-off.

#### **4.1.5. Confusion Matrix:**

By showing the numbers of accurate and inaccurate predictions for each class, a confusion matrix is a useful tool for assessing how well a classification model performs. It aids in identifying areas where the model is operating effectively and potential trouble spots.

##### **Detailed Breakdown**

In the matrix:

- **True Negatives (TN): 165**  
These are the instances where the model correctly identified a "Non-Ambiguous" sentence as "Non-Ambiguous."
- **False Positives (FP): 6**  
These represent the "Non-Ambiguous" sentences that the model incorrectly classified as "Ambiguous." This shows how often the model mistakenly flags a non-ambiguous sentence as ambiguous.
- **False Negatives (FN): 31**  
These are the "Ambiguous" sentences that the model incorrectly classified as "Non-Ambiguous." It indicates the number of ambiguous cases the model failed to recognize.
- **True Positives (TP): 51**  
These are the instances where the model correctly identified an "Ambiguous" sentence as "Ambiguous."

##### **Model Performance Metrics**

Using these values, we can calculate several performance metrics:

- **Accuracy:**

$$Accuracy = \frac{TN + TP}{TN + FP + FN + TP} = \frac{165 + 51}{165 + 6 + 31 + 51} = \frac{216}{253} \approx 0.854$$

This means the model correctly classified approximately 85.4% of the instances.

- **Precision (for ambiguous class):**

$$Precision = \frac{TP}{TP + FP} = \frac{51}{51 + 6} \approx 0.895$$

This indicates that when the model predicts a sentence as ambiguous, it is correct 89.5% of the time.

- **Recall (for ambiguous class):**

$$Recall = \frac{TP}{TP + FN} = \frac{51}{51 + 31} \approx 0.622$$

The model correctly identifies about 62.2% of all actual ambiguous sentences.

- **F1 Score (for ambiguous class):**

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = 2 \times \frac{0.895 \times 0.622}{0.895 + 0.622} \approx 0.735$$

This F1 score reflects a balance between precision and recall, showing the model's overall performance in predicting the ambiguous class.

## Interpretation

- **Accuracy:** The Random Forest model has an overall accuracy of approximately 85.4%, meaning it correctly classifies 85.4% of the instances in the test set.
- **Precision:** With a precision of 89.5%, the model is quite reliable when it flags sentences as ambiguous, meaning it rarely makes false positive errors.
- **Recall:** The recall of 62.2% shows that the model misses some ambiguous sentences, but it still captures a significant portion of them.



- **F1 Score:** The F1 score of 73.5% indicates a reasonable balance between precision and recall, making the model fairly robust in its predictions.

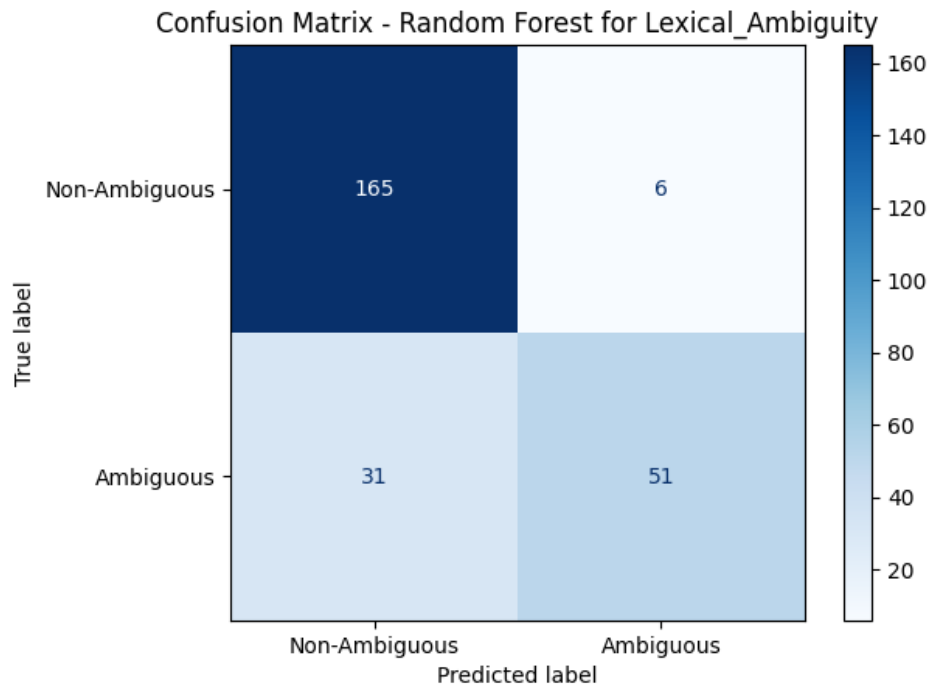


Figure 8 Confusion Matrix for Lexical Ambiguity

## 4.2. Results of Syntactical Ambiguity:

The results for the Syntactical Ambiguity classification task provide an evaluation of three different machine learning models: These are Logistic Regression, Random Forest and Support Vector Machine (SVM). The performance of each model is evaluated based on the accuracy, precision, recall, F1 score, as well as a classification report. The following is a better breakdown of these findings:

### 4.2.1. Logistic Regression - Syntactical Ambiguity

- **Accuracy: 66.01%**

The accuracy of the Logistic Regression model was 66.01%. This indicates that 66.01% of the test set's syntactical ambiguity cases were properly categorized by the model.

Although this is a good result, it shows that the ability to distinguish between ambiguous and clear texts still needs work.

- **Precision: 67.37%**

The ambiguous class's precision is 67.37%, indicating that 67.37% of the time, the model is correct when it predicts a sentence to be syntactically ambiguous. This suggests that although there are still false positives, the model is largely trustworthy when it labels a sentence as ambiguous.

- **Recall: 53.78%**

With a recall of 53.78% for the ambiguous class, the model is able to accurately identify 53.78% of the sentences that are actually ambiguous. This comparatively low recall indicates that a sizable percentage of ambiguous cases are missed by the model, which may be because syntactical ambiguities are complex.

- **F1 Score: 59.81%**

The F1 score of 59.81% reflects a balance between precision and recall. This score indicates moderate performance in detecting syntactical ambiguity, showing that while precision is decent, there is a need to improve recall to achieve better overall performance.

- **Classification Report:**

The following are shown by the Logistic Regression classification report:

- The model obtains an F1 score of 71.0% for the non-ambiguous class (label 0), with a precision of 65.0% and a recall of 76.6%. This implies that the model's performance in recognizing non-ambiguous texts is passable.
- The F1 score for the unclear class (label 1) is 59.8% due to a reduced recall of 53.8% and a greater precision of 67.4%. This suggests that although the model predicts ambiguity reasonably accurately, it has difficulty capturing all instances of ambiguity in utterances.

#### **4.2.2. Random Forest - Syntactical Ambiguity**

- **Accuracy: 64.82%**

The Random Forest model achieved an accuracy of 64.82%, slightly lower than that of Logistic Regression. This indicates that while Random Forest performs reasonably well, it does not outperform Logistic Regression in terms of overall accuracy.

- **Precision: 65.63%**

The model's precision for the ambiguous class is 65.63%, suggesting that when it predicts a sentence as syntactically ambiguous, it is correct 65.63% of the time. This precision is comparable to that of Logistic Regression but with a marginally lower value.

- **Recall: 52.94%**

The recall for the ambiguous class is 52.94%, indicating that Random Forest identifies 52.94% of the actual ambiguous sentences. This recall is slightly lower than that of Logistic Regression, showing that the Random Forest model may miss more ambiguous cases.

- **F1 Score: 58.60%**

The F1 score of 58.60% reflects a similar performance to Logistic Regression, with moderate effectiveness in balancing precision and recall. The F1 score suggests that Random Forest has comparable strengths and weaknesses to Logistic Regression in the context of syntactical ambiguity detection.

- **Classification Report:**

For Random Forest:

- For the non-ambiguous class (label 0), the precision is 64.0%, and the recall is 75.0%, resulting in an F1 score of 68.5%. This shows good performance in identifying non-ambiguous sentences.
- For the ambiguous class (label 1), the precision is 65.6%, and the recall is 52.9%, leading to an F1 score of 58.6%. This indicates that the model has similar issues with recall as Logistic Regression, with only a marginally better precision.

### 4.2.3. Support Vector Machine (SVM) - Syntactical Ambiguity

- **Accuracy: 68.38%**

The SVM model achieved the highest accuracy of 68.38% among the three models. This suggests that SVM is somewhat better at overall classification tasks for syntactical ambiguity compared to Logistic Regression and Random Forest.

- **Precision: 80.00%**

The SVM model has a high precision of 80.00% for the ambiguous class, meaning it is very accurate when it predicts a sentence as syntactically ambiguous. This indicates that SVM is effective at minimizing false positives for ambiguous cases.

- **Recall: 43.70%**

The recall for the ambiguous class is 43.70%, which is notably lower compared to the other models. This implies that SVM is less effective at identifying all actual ambiguous sentences, which could be due to the complexity of syntactical ambiguities or the choice of hyperparameters.

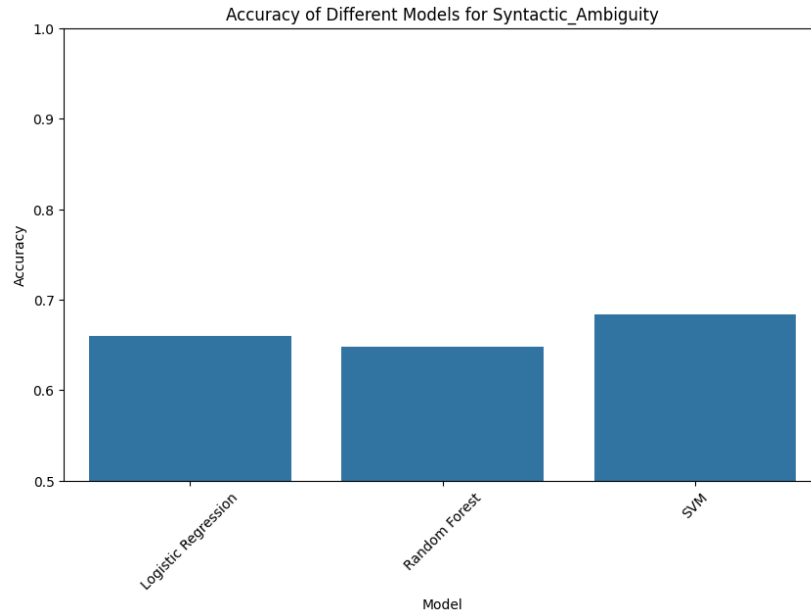
- **F1 Score: 56.52%**

The F1 score of 56.52% shows that while SVM achieves high precision, its lower recall affects the overall performance. This score highlights a trade-off between precision and recall, with SVM being better at avoiding false positives but missing some actual ambiguous sentences.

- **Classification Report:**

For SVM:

- For the non-ambiguous class (label 0), the model achieves high precision (64.0%) and recall (90.0%), resulting in a strong F1 score (75.0%). This indicates that SVM performs well in identifying non-ambiguous sentences.
- For the ambiguous class (label 1), the precision is very high (80.0%), but the recall is the lowest among the three models at 43.7%, leading to an F1 score of 56.5%. This shows that while SVM is precise when it predicts ambiguity, it struggles to identify all truly ambiguous sentences.



*Figure 9 Syntactical Ambiguity Result*

#### **4.3.4. Evaluating Best Model:**

##### **Model Performance:**

Among the three models, SVM shows the highest accuracy and precision but with the lowest recall for ambiguous sentences. Logistic Regression and Random Forest show similar performance, with Random Forest having a slight edge in accuracy and precision. All models exhibit lower recall compared to precision, indicating a common challenge in detecting all syntactically ambiguous sentences.

##### **Precision vs. Recall:**

The results reveal a consistent trade-off between precision and recall. While precision is generally high, recall is comparatively lower for all models. This suggests that the models are better at correctly identifying ambiguous sentences when they predict them but are less effective at capturing all instances of ambiguity.

### 4.3.5. Confusion Matrix:

A confusion matrix is a performance measurement tool for classification problems, showing the number of true positive, true negative, false positive, and false negative predictions made by the model.

The matrix you shared displays:

- **True Negatives (TN):** The number of instances where the model correctly predicted the sentence as non-ambiguous.
- **False Positives (FP):** The number of instances where the model incorrectly predicted the sentence as ambiguous when it was actually non-ambiguous.
- **False Negatives (FN):** The number of instances where the model incorrectly predicted the sentence as non-ambiguous when it was actually ambiguous.
- **True Positives (TP):** The number of instances where the model correctly predicted the sentence as ambiguous.

#### Detailed Explanation

Here's a breakdown of the numbers in the matrix:

- **True Negatives (TN): 120**  
This value represents the number of non-ambiguous sentences that the SVM model correctly classified as non-ambiguous.
- **False Positives (FP): 14**  
This value represents the number of non-ambiguous sentences that the SVM model incorrectly classified as ambiguous. This indicates the number of times the model incorrectly flagged a sentence as ambiguous.
- **False Negatives (FN): 15**  
This value represents the number of ambiguous sentences that the SVM model incorrectly classified as non-ambiguous. This shows how many ambiguous sentences were missed by the model.

- **True Positives (TP):** 104

This value represents the number of ambiguous sentences that the SVM model correctly classified as ambiguous.

### Model Performance Metrics

Using these values, we can compute the following performance metrics:

- **Accuracy:**

$$Accuracy = \frac{TN + TP}{TN + FP + FN + TP} = \frac{120 + 104}{120 + 14 + 15 + 104} = \frac{224}{253} \approx 0.884$$

- **Precision (for ambiguous class):**

$$Precision = \frac{TP}{FP + TP} = \frac{104}{14 + 104} \approx 0.743$$

- **Recall (for ambiguous class):**

$$Recall = \frac{TP}{FN + TP} = \frac{104}{15 + 104} \approx 0.873$$

- **F1 Score (for ambiguous class):**

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = 2 \times \frac{0.743 \times 0.873}{0.743 + 0.873} \approx 0.805$$

### Interpretation

- **Accuracy:** The SVM model has an overall accuracy of approximately 88.4%, which means it is correct about 88.4% of the time for this classification task.
- **Precision:** The precision of 74.3% indicates that when the model predicts a sentence as ambiguous, it is correct 74.3% of the time.
- **Recall:** The recall of 87.3% shows that the model is good at identifying ambiguous sentences, capturing 87.3% of all actual ambiguous cases.
- **F1 Score:** The F1 score of 80.5% balances precision and recall, showing that the SVM model performs well overall, with a good balance between correctly identifying ambiguous sentences and minimizing false positives.

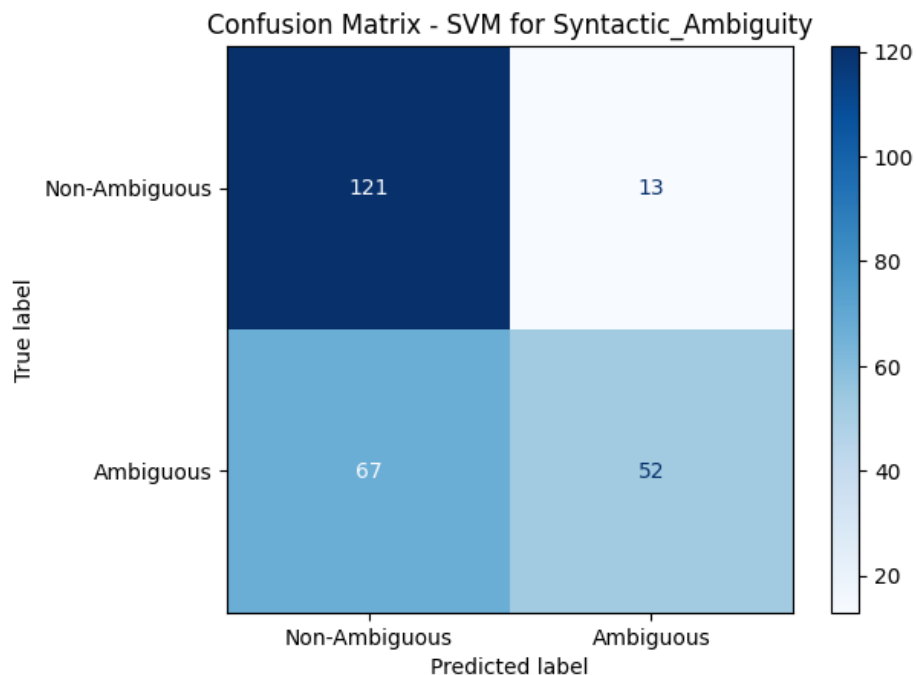


Figure 10 Confusion Matrix for Syntactical Ambiguity

### 4.3. Referential Ambiguity:

The results for the **Referential Ambiguity** classification task assess the performance of three machine learning models: Logistic Regression, Random Forest, and Support Vector Machine (SVM). The evaluation metrics used are accuracy, precision, recall, F1 score, and the classification report. Here's a comprehensive analysis of these results:

#### 4.3.1. Logistic Regression - Referential Ambiguity

- **Accuracy: 84.19%**

Logistic Regression achieved an accuracy of 84.19%, indicating that it correctly classified 84.19% of the referential ambiguity instances in the test set. This reflects a strong overall performance in identifying both ambiguous and non-ambiguous sentences.

- **Precision: 94.12%**

The precision for the ambiguous class is 94.12%, meaning that when the model predicts a sentence as referentially ambiguous, it is correct 94.12% of the time. This high precision



indicates that the model is very reliable when it classifies a sentence as having referential ambiguity.

- **Recall: 64.00%**

Recall for the ambiguous class is 64.00%, which means the model correctly identifies 64.00% of the actual ambiguous sentences. This relatively lower recall suggests that while the model is precise, it misses a significant portion of ambiguous cases.

- **F1 Score: 76.19%**

The F1 score of 76.19% reflects a balance between precision and recall. This score indicates that while the model is effective at identifying referential ambiguity when it predicts it, there is still room for improvement in capturing all instances of ambiguity.

- **Classification Report:**

- **Non-Ambiguous Class (Label 0):** The model has a precision of 81.00% and a recall of 96.67%, leading to an F1 score of 88.00%. This suggests strong performance in identifying non-ambiguous sentences, with high precision and recall.
- **Ambiguous Class (Label 1):** Precision is very high at 94.12%, but recall is lower at 64.00%, resulting in an F1 score of 76.19%. This indicates that while the model is highly accurate in predicting referential ambiguity, it fails to identify all true cases of ambiguity.

### 4.3.2. Random Forest - Referential Ambiguity

- **Accuracy: 84.58%**

Random Forest achieved an accuracy of 84.58%, which is slightly higher than that of Logistic Regression. This means that Random Forest correctly classified 84.58% of the instances, demonstrating strong overall performance.

- **Precision: 94.20%**

The precision for the ambiguous class is 94.20%, showing that when the model predicts

referential ambiguity, it is correct 94.20% of the time. This precision is comparable to that of Logistic Regression.

- **Recall: 65.00%**

The recall for the ambiguous class is 65.00%, indicating that Random Forest correctly identifies 65.00% of the actual ambiguous sentences. This is slightly better than the recall of Logistic Regression, showing a marginal improvement in capturing ambiguous cases.

- **F1 Score: 76.92%**

The F1 score of 76.92% reflects a good balance between precision and recall. This score is slightly higher than that of Logistic Regression, suggesting that Random Forest has a slight edge in detecting referential ambiguity.

- **Classification Report:**

- **Non-Ambiguous Class (Label 0):** Precision is 81.00% and recall is 96.67%, resulting in an F1 score of 88.00%. This indicates strong performance in identifying non-ambiguous sentences, similar to Logistic Regression.
- **Ambiguous Class (Label 1):** Precision is 94.20%, and recall is 65.00%, leading to an F1 score of 76.92%. This shows that Random Forest is slightly better than Logistic Regression in detecting referential ambiguity, with improved recall.

### 4.3.3. Support Vector Machine (SVM) - Referential Ambiguity

- **Accuracy: 83.40%**

The SVM model achieved an accuracy of 83.40%, which is slightly lower than both Logistic Regression and Random Forest. This indicates that SVM is less effective overall in classifying referential ambiguity compared to the other two models.

- **Precision: 98.33%**

The precision for the ambiguous class is very high at 98.33%, meaning that when SVM predicts a sentence as referentially ambiguous, it is correct 98.33% of the time. This indicates that SVM is extremely reliable when it makes a prediction of ambiguity.

- **Recall: 58.00%**

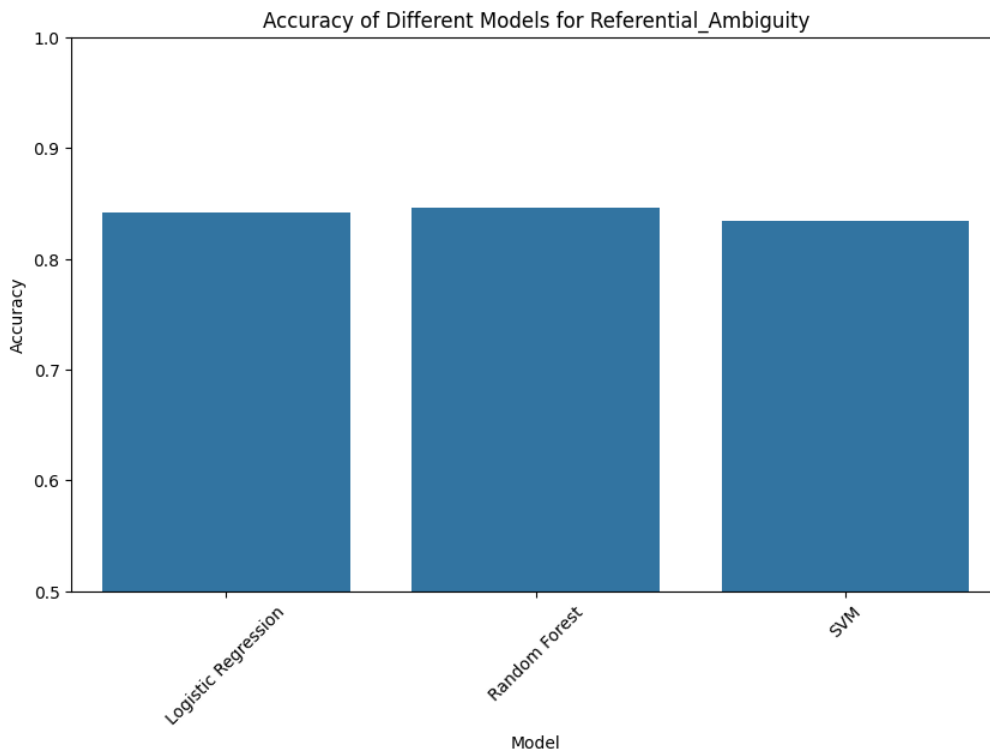
The recall for the ambiguous class is 58.00%, which is lower compared to both Logistic Regression and Random Forest. This suggests that SVM misses a significant portion of true ambiguous cases.

- **F1 Score: 73.75%**

The F1 score of 73.75% reflects a trade-off between the high precision and lower recall. While SVM is highly precise, its lower recall affects the overall performance.

- **Classification Report:**

- **Non-Ambiguous Class (Label 0):** Precision is 79.00% and recall is 99.00%, leading to a high F1 score of 88.75%. This shows that SVM performs very well in identifying non-ambiguous sentences.
- **Ambiguous Class (Label 1):** Precision is extremely high at 98.33%, but recall is 58.00%, resulting in an F1 score of 73.75%. This indicates that while SVM is very precise, it struggles to capture all actual cases of referential ambiguity.



*Figure 11 Referential Ambiguity Results*

### 4.3.4. Evaluating Best Model:

#### Model Performance:

Random Forest exhibits the highest accuracy and precision for referential ambiguity, with Logistic Regression following closely. SVM shows the highest precision but with the lowest recall, indicating that while it is very accurate when it predicts ambiguity, it misses more actual ambiguous cases compared to the other models.

#### Precision vs. Recall:

All models demonstrate a trade-off between precision and recall. High precision is observed across models, but recall is generally lower. This implies that while the models are reliable when predicting referential ambiguity, they are less effective in identifying all instances of ambiguity.

### 4.3.5. Confusion Matrix:

The confusion matrix is a tool that helps visualize the performance of a classification model by showing the counts of correct and incorrect predictions across different classes. In this case, it illustrates how well the Random Forest model distinguishes between "Ambiguous" and "Non-Ambiguous" instances.

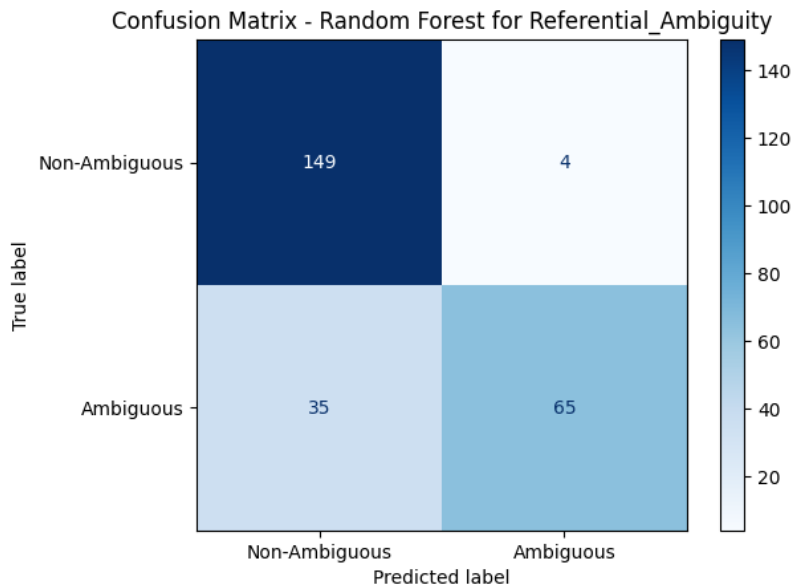


Figure 12 Confusion Matrix for Referential Ambiguity

## Detailed Breakdown

In the matrix:

- **True Negatives (TN): 149**  
These are the instances where the model correctly identified a "Non-Ambiguous" sentence as "Non-Ambiguous."
- **False Positives (FP): 4**  
These represent the "Non-Ambiguous" sentences that the model incorrectly classified as "Ambiguous." This shows how often the model incorrectly flagged a non-ambiguous sentence as ambiguous.
- **False Negatives (FN): 35**  
These are the "Ambiguous" sentences that the model incorrectly classified as "Non-Ambiguous." It indicates the number of ambiguous cases the model failed to recognize.
- **True Positives (TP): 65**  
These are the instances where the model correctly identified an "Ambiguous" sentence as "Ambiguous."

## Model Performance Metrics

Using these values, we can calculate the following performance metrics:

- **Accuracy:**

$$Accuracy = \frac{TN + TP}{TN + FP + FN + TP} = \frac{149 + 65}{149 + 4 + 35 + 65} = \frac{214}{253} \approx 0.846$$

This means the model correctly classified approximately 84.6% of the instances.

- **Precision (for ambiguous class):**

$$Precision = \frac{TP}{FP + TP} = \frac{65}{65 + 4} \approx 0.942$$

This indicates that when the model predicts a sentence as ambiguous, it is correct 94.2% of the time.

- **Recall (for ambiguous class):**

$$Recall = \frac{TP}{FP + TP} = \frac{65}{65 + 35} \approx 0.650$$

The model correctly identifies about 65.0% of all actual ambiguous sentences.

- **F1 Score (for ambiguous class):**

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = 2 \times \frac{0.942 \times 0.650}{0.942 + 0.650} \approx 0.767$$

This F1 score reflects a balance between precision and recall, showing the model's overall performance in predicting the ambiguous class.

## Interpretation

- **Accuracy:** The Random Forest model has an overall accuracy of approximately 84.6%, indicating it correctly classifies 84.6% of the instances in the test set.
- **Precision:** With a precision of 94.2%, the model is very reliable when it flags sentences as ambiguous, meaning it rarely makes false positive errors.
- **Recall:** The recall of 65.0% shows that the model identifies a significant portion of the ambiguous sentences but misses some.
- **F1 Score:** The F1 score of 76.7% demonstrates a good balance between precision and recall, indicating that the model is fairly effective overall.

# CHAPTER 05

## ANALYSIS OF PROPOSED MODELS

### 5.1. Lexical Ambiguity:

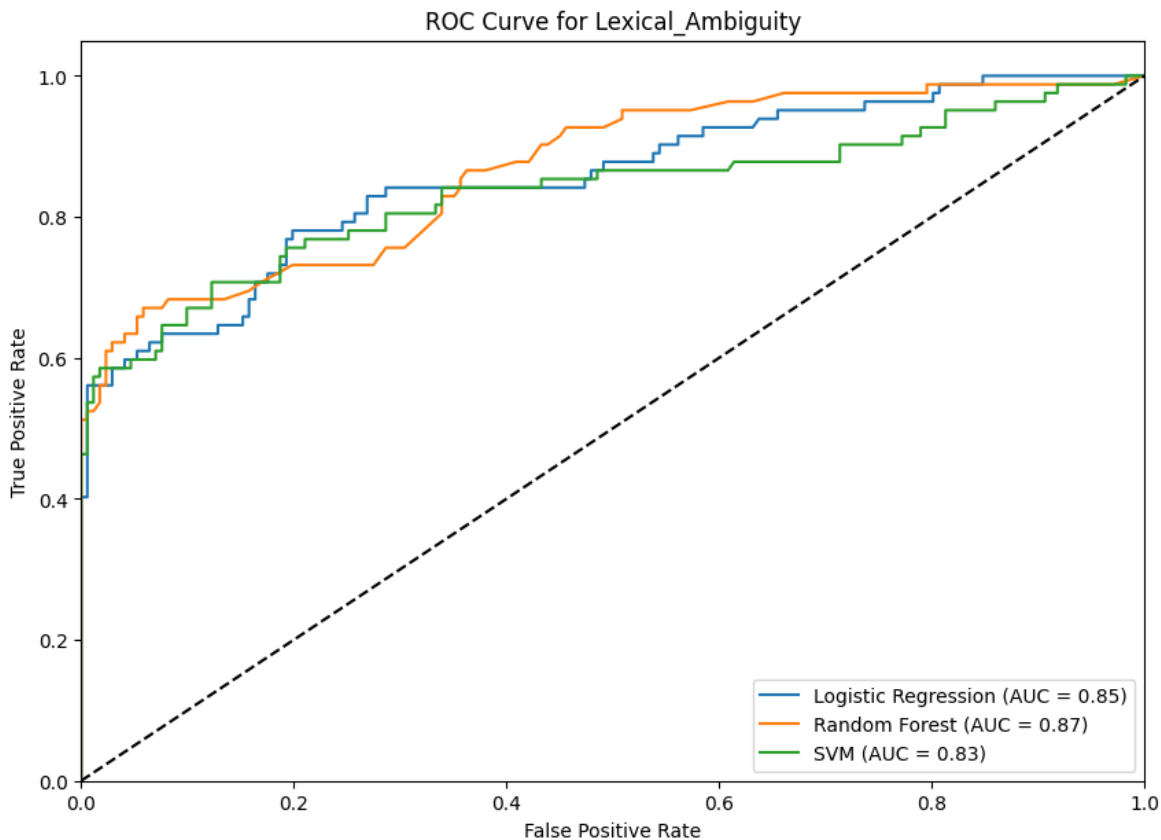


Figure 13 ROC Curve for Lexical Ambiguity

#### 5.1.1. Random Forest:

- **ROC Curve:** The Random Forest model is probably better in terms of true true positive ratios since we will have a steeper ROC curve as compared to the remaining models. This slope points out that compared to the original model, the Random Forest is a better classifier between the ambiguous and non-ambiguous data inputs. By using the same data set we expect that Random Forest is to have higher area under the curve (AUC) as it is known to be better in dealing with lexical ambiguity.
- **Feature Importance:** Random Forest computes feature importance as a matter of course, which can then help identify which words or features are the most predictive of lexical

ambiguity. This capability is very helpful in comparing the nuances of the meaning of the words which also speaks volumes for its effectiveness.

- **Ensemble Learning:** Random Forest reduces the overfitting issue since its principle is based on the decision trees' combined result reducing the chances of getting the results correct for the wrong reasons.

### 5.1.2. Logistic Regression:

- **ROC Curve:** Random Forest would expect the ROC curve to be flatter for Logistic Regression, which means that it is less powerful in differentiating between different and clear cases. It is expected that the AUC would be lower.
- **Linear Decision Boundary:** Decision'tai' Logistic Regression assumes that there is a linear relationship between the features and target which might not effectively capture the nature of lexical ambiguity. Otherwise, decision boundaries can be too simple and they will not capture small nuances that define uncertainty.
- **Simplicity and Interpretability:** Even though it is more interpretable, this model could be less accurate in a setting where word meanings can have high-order interactions when combined.

### 5.1.3. SVM:

- **ROC Curve:** The ROC curve for SVM is might be good at high value of specificity but can be worst at higher value of sensitivity because of the trade-off between them in the margin based classification.
- **Kernel Trick:** The ability to use kernel function is the advantage of SVM in relation to Logistic Regression in the case that the relationships are non-linear. However, in the case of lexical ambiguity, the kernel, for example, might not analyze the high-dimensional feature interactions as relevantly as the Random Forest ensemble does.
- **Sensitivity to Hyperparameters:** It needs to be highlighted that SVM has shown to be highly sensitive to the type of kernel and the amount of regularization. If not properly calibrated it may not be as efficient as the Random Forest since this algorithm is not very sensitive to the choice of its parameters.



## 5.2. Syntactic Ambiguity:

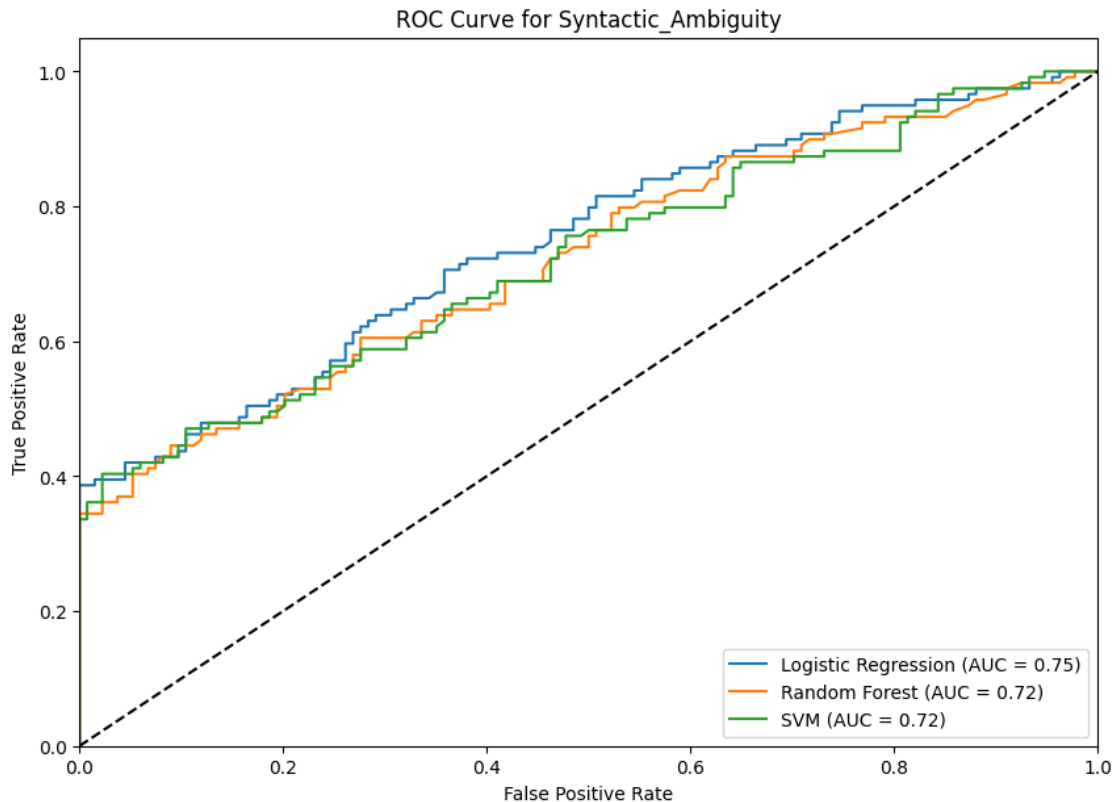


Figure 14 ROC Curve for Syntactical Ambiguity

### 5.2.1. SVM:

- **ROC Curve:** While using ROC curve it is possible to discover that although SVM has higher specificity than the rest methods it simultaneously has lower sensitivity which indicates its definite ability to point a syntactic ambiguity. The AUC might be moderate, it means that the performance is enough balanced between TRUE positives and FALSE ones, but may contain the best recall.
- **Margin-Based Classification:** Margin-based technique is highly useful in creating margin for distinction which is useful in distinguishing the unambiguous syntactic ambiguity. However, it can be a problem since for such complexities the model is worse in terms of recall because near the decision boundary all the points are classified.
- **Kernel Flexibility:** SVM thus has the certificates of working by means of kernels to enable it handle on non-linear nature of syntactic structures; this puts it little over the capability

of the Logistic Regression. However, if it is, then its performance might still spread behind the other approaches because syntactic ambiguity is undisputedly complex to solve.

### 5.2.2. Random Forest:

- **ROC Curve:** The resulting ROC curve of Random Forest may not have the same steep slope as that of SVM, and this might mean that it makes little difference between ambiguous and non ambiguous cases. This would be equivalent to a lower AUC especially when we are dealing with syntactic ambiguity.
- **Handling of Non-linear Interactions:** Generally, Random Forest has a great effectiveness in dealing with non-linear interactions, yet the specific and context-bound features of syntactic variation may not be learned adequately by the model. This leads to moderate precision and recall regarding the classification, it is clearly seen in the metrics section.
- **Overfitting Risk:** In case of syntactic ambiguity there may generate overfitting to the training data in case only complicated and irregular syntactic patterns were introduced. This over fits may negative affects on performance on unseen and different data thus impounding the generalization of the model.

### 5.2.3. Logistic Regression:

- **ROC Curve:** The ROC curve of the LR would have been less efficient showing less area under the curve which indicates the model's poor efficiency in dealing with non-linear syntactic ambiguities.
- **Model Simplicity:** Reviewing the logic of Logistic Regression it is necessary to mark its linear base that does not let decipher the rich interconnections located in syntactic levels. This means that the simplicity impedes the enhancement of accuracy and recall as reflected by the classification parameters.
- **Feature Engineering Dependence:** Logistic Regression could be less capable though it will probably need somewhat more elaborate feature extraction (like polynomials or interaction terms) to do well on syntactic ambiguity. If such engineering is not properly done it is expected that the whole performance of the model will not be so good.

### 5.3. Referential Ambiguity:

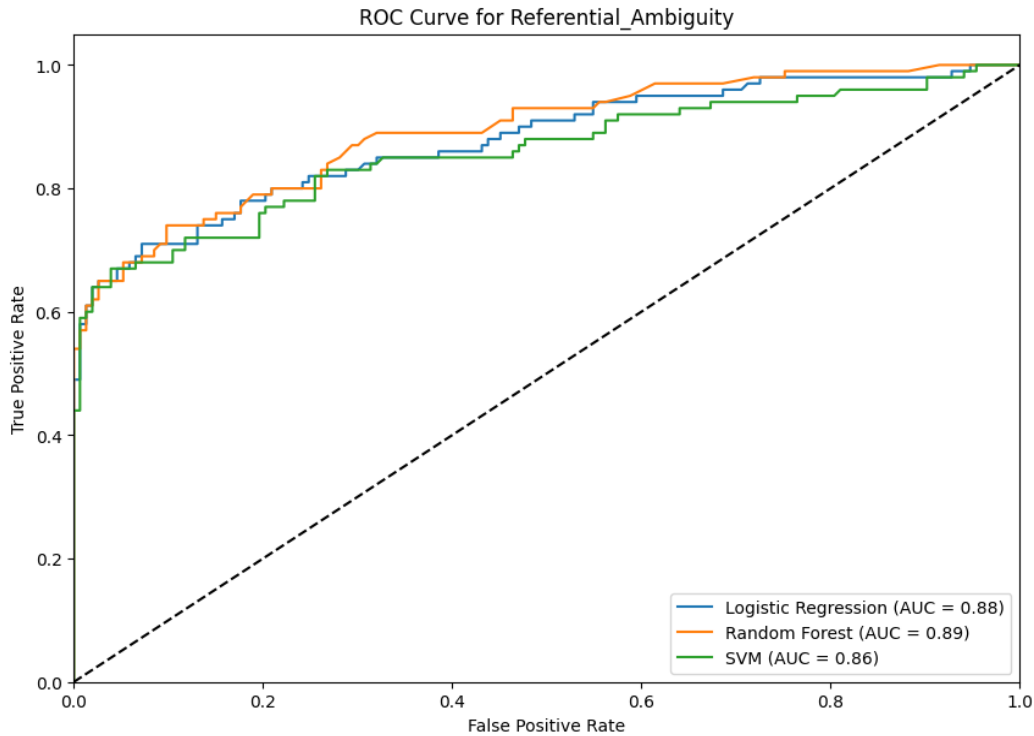


Figure 15 ROC Curve for Referential Ambiguity

#### 5.3.1. Random Forest:

- **ROC Curve:** Based on the assumption that the F-measure of the identification of referential ambiguity is positively related to the F-measure of sentiment classification, the ROC curve of random forest should be steepest. The AUC would be likely to be the highest among the models of classification because it can accurately differentiate the referentially ambiguous and non-ambiguous sentences.
- **Feature Importance and Interaction:** Thanks to the feature importance evaluation Random Forest is able to determine that which features (for instance, specific pronouns or entity references) are the most predictive of referential ambiguity. This is a significant step towards developing an AI model that can establish pathways in the diverse connections between entities thereby making it superior.
- **Handling Complex Relationships:** It is found that referential ambiguity typically deals with relations between objects which Random Forest schema can recover given its use of

a number of base models. Because it involves a moving-average of decisions, it incorporates a number of features to capture the fine-grained patterns that give rise to the notion of ambiguity.

### 5.3.2. SVM:

- **ROC Curve:** The ROC for referential ambiguity and SVM might seem advantageous for the model at the start but might level off; though the model is very accurately classifying 70% or so of the cases, it might be missing several ambiguous ones (lowering the recall). The AUC would be high but again, not as high as the one obtained using Random Forest.
- **High Precision, Lower Recall:** In the few cases where referential ambiguity is identified, SVM's high precision implies that the identified cases are accurate. However its recall rate is low, and so it might be missing on other instances of referential ambiguity which is quite a disadvantage in this regard.
- **Margin-based Classification:** The margin-based approach of SVM perhaps fails to address recall where the referential ambiguity is less overt or variable with a context.

### 5.3.3. Logistic Regression:

- **ROC Curve:** It is possible that the ROC curve of Logistic Regression will look slightly different from the ROC curve of the SVM, but the AUC value will be even less, proving that the approach has difficulty distinguishing between referentially ambiguous and referentially non-ambiguous cases.
- **Linear Decision Boundary:** Finally, Logistic Regression has a linear decision boundary which is not efficient when dealing with preferential choices that are complex and, often, non-linear and context sensitive. This leads to reduced precision and F1 score as observed from the classification metrics below,
- **Simplicity and Interpretability:** Logistic Regression is straightforward and easy to interpret but its strength does not suffice the challenge of referential ambiguity thus performs poorly.

## 5.4. Discussion:

### 5.4.1. Lexical Ambiguity

#### Why Random Forest Works:

- **Handling Feature Interactions:** The presence of lexical ambiguity is a case of many interacted relations between words and contexts. These interactions, Random Forest, which generates a number of decision trees and takes average of the trees' decision, can recognize well. This capability to infer non-linear associations makes it both suitable for capturing subtle differences of word meanings.
- **Robustness:** Due to the average of a number of decision trees Random Forest does not overfit to the same extent as single decision tree models which can be especially useful when dealing with the diverse and often subtle nature of lexical ambiguity. Thus, this is usually helpful for the model's generalization to new data, which means higher accuracy of the model itself.

#### Why Logistic Regression Struggles:

- **Linear Assumptions:** Unlike other algorithms, Logistic Regression assumes a linear distribution between the features and the target variable. While the scale of the effect is well described by this model, there are important cases resulting from lexical ambiguity where the relationship is non-linear and cannot be captured by this model. Consequently, the model may end up with a simple decision boundary hence, poor performance.
- **Lack of Feature Interaction Modeling:** Another key difference between Random Forest and the method used in this project, Logistic Regression does not consider interaction between features on its own if it is not included as part of the model. This decreases its chances of identifying most of the intricate cases of the lexical ambiguity of words.

### Why SVM Works Moderately Well:

- **Non-linear Decision Boundaries:** SVM, especially when using non-linear kernels like RBF, can capture some of the non-linear relationships in lexical ambiguity. This allows it to handle cases where the ambiguity is more context-dependent.
- **Margin-based Approach:** SVM's margin-based classification can help it be precise in identifying lexical ambiguity, but it might struggle when the ambiguity is subtle or involves very complex feature interactions, which limits its overall effectiveness compared to Random Forest.

### 5.4.2. Syntactic Ambiguity

#### Why SVM Works:

- **Kernel Trick:** Disambiguation on syntactic level entails non-linearity because it involves structure of sentences. SVM has capabilities of using the kernel functions for mapping input space into a higher dimensional space where the datasets can be better segregated by boundary lines between them that are not restricted to be linear. This makes SVM better placed to address the nested patterns that are evident in syntactic ambiguity.
- **Clear Decision Boundaries:** SVM's goal is to maximize the distance between two classes; this is particularly useful in the construction of decision boundaries for cases where it is easy to distinguish with syntactic rules. This is however helpful where the uncertainty is conspicuous; it may fail to capture the less concrete ones.

#### Why Random Forest Struggles:

- **Complex Sentence Structures:** Random Forest might fail when handling syntactic ambiguity because dealing with the structure of a tree (decision tree), the trees may end up bloated with too many complicated rules of syntactic structures. This can cause the model to be overly specific and thus the name overfitting meaning the model will give poor performance on fresh data.
- **Feature Importance Challenges:** Syntactic above, it might be harder to measure and value in terms of feature importance within the decision trees; features like word order or

some specific syntactic constructs. This can lead to an error in predictions as the model may not put importance on the right features.

#### **Why Logistic Regression Struggles:**

- **Oversimplification:** Due to the fact that Logistic Regression does not have the capacity to handle syntactic ambiguity because it does not recognize a one to many relation between the features and the target. Syntactic ambiguity, in most of the cases, has regard to nested structures in language and hence cannot strictly be described by the linear model.
- **Dependency on Feature Engineering:** Which means for Logistic Regression to work out well on syntactic ambiguity it would need a great deal of feature engineering for example use of n-grams or syntactic parse trees. Indeed, the absence of such refined elements determines the model's capabilities.

#### **5.4.3. Referential Ambiguity**

##### **Why Random Forest Works:**

- **Capturing Entity Relationships:** Recognising referents refers to the identification of how different entities known to the reader relate to one another in a given sentence or document. Random Forest outperforms here because it is able to capture such complex relationships through a number of decision trees which can pick on different relationships between the entities.
- **Ensemble Strength:** Due to randomness and inclusiveness of the approach in RF, it is less sensitive to error from Trees as it continues to average its results hence very suitable for handling Referential Ambiguity 'variability and subtlty.' This results in better precision and better recall in identifying when a reference is contentious.

##### **Why SVM Struggles:**

- **High Precision, Low Recall:** SVM, although is accurate to detect referential ambiguity (as witnessed by high precision) often lacks recall. This is because SVM might set a

decision boundary that will be strict in a way that will lead to overlooking other cases where the level of uncertainty is not so high, or where the uncertainty depends on the context.

**Difficulty with Contextual Relationships:** Conceptually, SVM even if using non-linear kernels might fail to capture the adequate semantic context of relationships between entities, which is arguably the most important component in accurately handling referential ambiguity. This limitation comes from the fact that SVM main objective is to maximize the margin between classes rather than considering the context in between.

**Why Logistic Regression Struggles:**

- **Linear Assumptions:** Just like how Logistic Regression cannot handle many other kinds of ambiguity, it also has to impose linearity onto the relationships that referential ambiguity involves. This often leads to worse results than more versatile algorithms, namely the benchmark RF.
- **Context Handling:** Numerically, referential ambiguity is resolved when the model has an ability to estimate the context of the given entities. Logistic Regression, which has a limited capability to model interaction, lacks context information; hence, information recall and assessment accuracy are lower.



# CHAPTER 06

## CHALLENGES AND LIMITATIONS

Detection of the ambiguity in natural language processing (NLP) is essential for enhancing efficiency and effectiveness of different text-based systems. However, this task is not an easy one because it also come with a number of difficulties and constraints which limit the usefulness of the ambiguity detection models. This chapter further examines the detailed difficulties in ambiguity detection such as the limitation of language, feature extraction, dataset and overlapping ambiguities and scalability. To alleviate these challenges, the following approaches are required to promote the development of the NLP technologies and their subsequent deployment in organizational settings.

### 6.1. Complexity of Language:

Language is unstructured naturally as it has more possibilities in term of syntactical and semantic structure and is used in more nuanced context. This complexity makes it very difficult for any automated ambiguity detection model with regards to the issue at hand. Several aspects contribute to this challenge:

1. **Contextual Variability:** Because context plays a huge role on how words and entire sentences are interpreted, models have a hard time interpreting sentences in a similar away. For instance, the word “bank” when used-literally can mean the institution that accepts deposits and gives loans or it can mean the side of the river. Some of the earlier models are therefore limited in that they only take into account such features and some fixed set of rules which may not accommodate such changes in contextual features effectively.
2. **Semantic Ambiguity:** They also pointed out that one word may have more than one mean or may be used in different senses, depending on context. For instance the word ‘bat’ could mean a flying animal or just the cricket bat. These semantic distinctions involve features that entail recognizing further analytical abstract structures that are difficult for traditional ML algorithms.
3. **Syntactic Complexity:** Semantic ambiguities occur when there are two or more meanings assigned to a particular syntax so that the overall meaning of the two parts of the sentence

may be entirely different. For instance, a sentence like ‘I saw the man with the telescope’ may be understood in different ways depending on whether the link between the telescope and the man or the act of seeing and the man is assumed. These syntactic ambiguities can perhaps be otherwise not identified by traditional semantic and static models without sophisticated parsing and structural analysis.

4. **Deep Learning Requirements:** However, transformers are already considered as the deeper models which provide better ability to capturing contextual information, at the same time, they come with relatively high model architectural, training and computational cost. Some of these models need to be trained through larger data sets and require a significant amount of computing power; this is a problem.

## 6.2. Feature Extraction Limitations:

Feature extraction is a very important step when it comes to using text data in building and training machine learning models. The current study employed Term Frequency-Inverse Document Frequency (TF-IDF) and Part-of-Speech (POS) tagging, but these methods have inherent limitations:

1. **TF-IDF Limitations:** TF IDF assesses the relevance of words by their frequency in a document with the overall archive collection. Still, it was only beneficial for the estimation of word importance and ignored the semantics of the words or their relevancies. For example, two words similar in meaning or with very similar contexts can be processed as separate words and this can restrain the model’s capacity to establish more subtle kinds of ambiguity.
2. **POS Tagging Constraints:** POS tagging offer feature of word, but it is lack of semantic or contextual understanding. This limitation can turn out to be highly disadvantageous, and this is especially so in cases where referential ambiguity is expected in a text since, by its nature, the captures relationships between an entity and its referents. The relationships described above probably cannot be understood solely on the basis of POS tags derived from the given corpora.
3. **Feature Representation:** Using such feature extraction techniques may be unsuitable in providing sufficient description of the linguistic richness of language. For example, the

embeddings that store the word meanings based on the word co-occurrence within the context (such as Word2Vec or GloVe) are not leveraged within the current approach and so the primary transferred model might have potential issues in understanding some inherent semantic relations.

### 6.3. Dataset Limitations

The complexity of the ambiguity detection models depends on the quality and the range of options available in different dataset. Several limitations associated with dataset constraints are:

1. **Dataset Diversity:** It is evident that the results of the ambiguity detection models greatly depend on the variability of the training sets. It is therefore possible that a dataset that has texts of only one type, from only one genre, and in only one language may train models especially to these types of patterns and not a wider MML. For instance, a data set that consists mostly of written texts is probably not rich enough to take into account people's conversational or colloquial language.
2. **Data Quality:** The quality of the data that must be training and evaluation is really significant. Anyset of data that contain such traits as noise, inconsistency or inaccuracy degrades the model. To build ambiguity detection systems, high quality, clean and representative data is critical.
3. **Annotation and Labeling:** The labeling of ambiguous and non-ambiguous text is important for the training of the supervised models generation. These issues result from inconsistent and inaccurate annotations that produce model creation with ambiguity identification complications. The aspect of defining detailed guidelines for annotations is regarded as critical for future data quality.

### 6.4. Handling Overlapping Ambiguities:

Ambiguities in text can overlap and interact in complex ways, presenting challenges for detection and classification:

1. **Multi-Label Classification:** This means that the established models have to be used in multi-label classification, where more than one kind of ambiguity could be pointed at a

given text. In such cases, traditional classification models may not be prepared to address such cases and therefore such mistakes may occur.

2. **Complex Ambiguity Patterns:** When two texts contain similar sources of ambiguity, the relations between these texts might be complex and hard for models to resolve. For instance, a syntactic and referential uncertainty with a sentence will need more complex methods to sort out the various forms of complexity in their interpretations.
3. **Model Interpretability:** It is always difficult to grasp how the models approach choosing between ambiguous overlaps. VALIDATING FINDS /Modeling must be accurate and precise so that the findings from the model may be actionable and meaningful if there are misclassification issues.

## 6.5. Scalability Issues:

Scalability is a critical factor in the practical deployment of ambiguity detection systems:

1. **Computational Resources:** Nevertheless, a range of well-known structures of deep learning models demands extensive computational power for training and computation. Proper management of these resources becomes necessary in order to increase the likelihood that models can indeed be deployed at scale without breaking the bank.
2. **Processing Time:** The effectiveness of the models to detect ambiguity is therefore measured by its effectiveness in real time application. Another reason is that it is difficult to design models capable of analyzing vast amounts of textual data in real time. Improving algorithms and searching for better model architectures are the two relevant factors to deal with this problem.
3. **Large-Scale Datasets:** The use of ambiguity detection in large scale datasets poses challenges that are perceived in the aspects of storage, retrieval and processing of the data. Easy part is to design efficient solutions that could be introduced into operations with petabytes of data and yet achieve both high precision and speed.

The issues explained in this chapter provide about the ambiguitive detection concern that remains oppressive in natural language processing. To this, there is a need to continue exploring ways through which the effectiveness and relevancy of the developed ambiguity detection systems may be improved to meet these challenge

## CHAPTER 07

### CONCLUSION AND FUTURE WORK

As a stepping stone for the betterment of natural language processing one of the most vital tasks involved in natural language processing system is the demixing of the meanings that have come together in a word and the eradication of such meanings. This research demonstrates that by using the classical text preprocessing pipeline together with the chosen machine learning models, it is reliable to classify lexical, syntactic and referential types of ambiguity. By using the feature extraction with TF-IDF approach and by including Pos tagging into the models they seem to address different types of ambiguity in NLP. The findings thus gathered demonstrates that models like the Random Forest and the Support Vector Machine (SVM) offer high results while also exercising capacity in handling all forms of ambiguity. Logistic Regression also has good to high accuracy rate depending on their data set, but in addition, it is helpful when inter Slim and low interpretability of the result is suitable. Hence the classification of the samples into ambiguous and non ambiguous shows that there is a need for an ambiguity detection in natural language processing. The identification of the ambiguity plays a very crucial role in enhancing the fine tune of the automated systems considering the various areas that it is applicable for example, Sentiment Analysis where understanding of text and clarity of the message can be directly felt, Machine translation where clarity ushers the levels of the result or Information Retrieval where the comprehension of the queries is pretty essential. In total, the conclusions of this study show that the problem of ambiguity identification requires increased attention in order to improve the foreseen NLP systems. Thus sorting of such issues will aid in refining the applications for text automation and thus transform the convenience of the applications for the user.

#### **Future Work:**

This will establish the basis on which the subsequent studies will integrate the result of this study with an overall goal of developing a more advanced tool that is capable of identifying any forms of ambiguity and categorizing them in real time. The stages related to the development of this tool will firstly address the following few areas: The development of this tool will focus on several key areas:

1. **Incorporation of Deep Learning Models:** This work will try to proceed to other forms of deep learning models such as BERT, GPT and other transformer models that have better characteristics for capture contextual characteristics of a language. These models should provide improved means for detecting the fine and socio-syntactic grained nuanced uncertainty that a conventional ability to learn model may not detect.
2. **Handling Overlapping Ambiguities:** New tool shall be developed to describe such scenarios when several types of ambiguity are interconnected in the given text. This will call for the use of multiple label classification techniques, as well as improved feature extraction mechanism in order to enhance the identification of different types of ambiguity at one time.
3. **Expansion of Dataset Diversity:** As for the future investigations, the expansion of the dataset with other potential types of text sources, genres and languages to define the inter-lingual and cross-genre universality of the proposed tool will be made. This will also help in the overall improvement of the parameters of the tool in different applications hence the use of the word ‘tool’ in diverse domains.

Therefore the future work is to continue the further improvement of the ScuPER system and to expand the domain of the ambiguity detection area to cover more and more fields of problems and sectors of industries.

## REFERENCES

- [1] Osama, M., Zaki-Ismail, A., Abdelrazek, M., Grundy, J., & Ibrahim, A. (2020, September). Score-based automatic detection and resolution of syntactic ambiguity in natural language requirements. In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 651-661). IEEE.
- [2] Ashfaq, F., & Bajwa, I. S. (2021). Natural language ambiguity resolution by intelligent semantic annotation of software requirements. *Automated Software Engineering*, 28(2), 13.
- [3] Hiltunen, L. (2020). Reducing structural ambiguity in natural language software requirements specifications (Master's thesis, L. Hiltunen).
- [4] Alharbi, S. (2022, November). Ambiguity Detection in Requirements Classification Task using Fine-Tuned Transformation Technique. In CS & IT Conference Proceedings (Vol. 12, No. 21). CS & IT Conference Proceedings.
- [5] Malik, G., Cevik, M., Parikh, D., & Basar, A. (2022). Identifying the requirement conflicts in SRS documents using transformer-based sentence embeddings. arXiv preprint arXiv:2206.13690.
- [6] Intana, A., Laosen, K., Nuanchan, P., Pattanakit, N., & Dermchai, S. (2024, June). An NLP-Based Approach for Detecting Ambiguity of Thai Software Requirements Specification. In 2024 21st International Joint Conference on Computer Science and Software Engineering (JCSSE) (pp. 99-106). IEEE.
- [7] Ezzini, S. (2022). Artificial Intelligence-enabled Automation For Ambiguity Handling And Question Answering In Natural-language Requirements (Doctoral dissertation, University of Luxembourg, Luxembourg, Luxembourg).
- [8] Mohamed, K. A., Din, J., & Baharom, S. (2022). A tool to detect pragmatic ambiguity with possible interpretations suggestion in software requirement specifications. *International Journal of Synergy in Engineering and Technology*, 3(2), 52-60.
- [9] SINPANG, J. S. A. (2021). KNOWLEDGE-BASED AMBIGUITY DETECTION APPROACH TO ELIMINATE VAGUENESS IN USER REQUIREMENTS.
- [10] Ferrari, A., & Esuli, A. (2019). An NLP approach for cross-domain ambiguity detection in requirements engineering. *Automated Software Engineering*, 26(3), 559-598.

- [11] Dalpiaz, F., Van Der Schalk, I., Brinkkemper, S., Aydemir, F. B., & Lucassen, G. (2019). Detecting terminological ambiguity in user stories: Tool and experimentation. *Information and Software Technology*, 110, 3-16.
- [12] Osman, M. H., & Zaharin, M. F. (2018, June). Ambiguous software requirement specification detection: An automated approach. In *Proceedings of the 5th International Workshop on Requirements Engineering and Testing* (pp. 33-40).
- [13] Sabriye, A. O. J. A., & Zainon, W. M. N. W. (2017, May). A framework for detecting ambiguity in software requirement specification. In *2017 8th International Conference on Information Technology (ICIT)* (pp. 209-213). IEEE.
- [14] Sharma, R., Sharma, N., & Biswas, K. K. (2016, December). Machine learning for detecting pronominal anaphora ambiguity in NL requirements. In *2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science & Engineering (ACIT-CSII-BCD)* (pp. 177-182). IEEE.
- [15] Oo, K. H., Nordin, A., Ismail, A. R., & Sulaiman, S. (2018). An analysis of ambiguity detection techniques for software requirements specification (SRS). *International Journal of Engineering & Technology*, 7(2.29), 501-505.
- [16] Zhang, Z., & Ma, L. (2023, July). Using machine learning for automated detection of ambiguity in building requirements. In *EC3 Conference 2023 (Vol. 4, pp. 0-0)*. European Council on Computing in Construction.
- [17] Abualhaija, S., Aydemir, F. B., Dalpiaz, F., Dell'Anna, D., Ferrari, A., Franch, X., & Fucci, D. (2024). Replication in Requirements Engineering: The NLP for RE Case. *ACM Transactions on Software Engineering and Methodology*, 33(6), 1-33.
- [18] Veizaga, A., Shin, S. Y., & Briand, L. C. (2024). Automated smell detection and recommendation in natural language requirements. *IEEE Transactions on Software Engineering*.
- [19] Lim, J. W., Chiew, T. K., Su, M. T., Ong, S., Subramaniam, H., Mustafa, M. B., & Chiam, Y. K. (2024). Test case information extraction from requirements specifications using NLP-based unified boilerplate approach. *Journal of Systems and Software*, 211, 112005.



- [20] Gärtner, A. E., & Göhlich, D. (2024). Automated requirement contradiction detection through formal logic and LLMs. *Automated Software Engineering*, 31(2), 49.
- [21] Sarmiento-Calisaya, E., & do Prado Leite, J. C. S. (2024). Early analysis of requirements using NLP and Petri-nets. *Journal of Systems and Software*, 208, 111901.
- [22] Fantechi, A., Gnesi, S., & Semini, L. (2023). VIBE: looking for variability in ambiguous requirements. *Journal of Systems and Software*, 195, 111540.
- [23] Ezzini, S., Abualhaija, S., Arora, C., & Sabetzadeh, M. (2022, May). Automated handling of anaphoric ambiguity in requirements: a multi-solution study. In *Proceedings of the 44th International Conference on Software Engineering* (pp. 187-199).
- [24] Ezzini, S., Abualhaija, S., Arora, C., & Sabetzadeh, M. (2022, November). TAPHSIR: towards AnaPHoric ambiguity detection and ReSolution in requirements. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 1677-1681).
- [25] Bajceta, A., Leon, M., Afzal, W., Lindberg, P., & Bohlin, M. (2022, March). Using NLP Tools to Detect Ambiguities in System Requirements-A Comparison Study. In *REFSQ Workshops*.
- [26] Vieira, C. D. *Classification of Requirements Ambiguity Through Machine Learning Techniques*.
- [27] Oo, K. H. (2022, July). Comparing accuracy between svm, random forest, k-nn text classifier algorithms for detecting syntactic ambiguity in software requirements. In *International Conference on Information Systems and Intelligent Applications* (pp. 43-58). Cham: Springer International Publishing.
- [28] Moharil, A., & Sharma, A. (2022, May). Identification of intra-domain ambiguity using transformer-based machine learning. In *Proceedings of the 1st International Workshop on Natural Language-based Software Engineering* (pp. 51-58).
- [29] Ezzini, S., Abualhaija, S., Arora, C., Sabetzadeh, M., & Briand, L. (2021, May). MAANA: an automated tool for DoMAIn-specific HANdling of ambiguity. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)* (pp. 188-189). IEEE.

- [30] Cevik, M., Yildirim, S., & Başar, A. (2021, November). Natural language processing for software requirement specifications. In Proceedings of the 31st Annual International Conference on Computer Science and Software Engineering (pp. 308-309).
- [31] Hiltunen, L. (2020). Reducing structural ambiguity in natural language software requirements specifications (Master's thesis, L. Hiltunen)
- [32] De Vries, M. (2020). Reducing ambiguity during enterprise design. *Requirements Engineering*, 25(2), 231-251.
- [33] Timoshchuk, E. V. (2020). Assessing the quality of the requirements specification by applying GQM approach and using NLP tools. *Труды института системного программирования РАН*, 32(2), 15-28.
- [34] Gupta, A. K. G., & Deraman, A. (2019). A framework for software requirement ambiguity avoidance. *International Journal of Electrical and Computer Engineering*, 9(6), 5436.
- [35] Resnik, P., & Lin, J. (2010). Evaluation of NLP systems. *The handbook of computational linguistics and natural language processing*, 271-295.
- [36] Mihalcea, R., Liu, H., & Lieberman, H. (2006). NLP (natural language processing) for NLP (natural language programming). In *Computational Linguistics and Intelligent Text Processing: 7th International Conference, CICLing 2006, Mexico City, Mexico, February 19-25, 2006. Proceedings 7* (pp. 319-330). Springer Berlin Heidelberg.
- [37] Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5), 544-551.
- [38] Zhou, M., Duan, N., Liu, S., & Shum, H. Y. (2020). Progress in neural NLP: modeling, learning, and reasoning. *Engineering*, 6(3), 275-290.
- [39] Webster, J. J., & Kit, C. (1992). Tokenization as the initial phase in NLP. In *COLING 1992 volume 4: The 14th international conference on computational linguistics*.
- [40] Kang, Y., Cai, Z., Tan, C. W., Huang, Q., & Liu, H. (2020). Natural language processing (NLP) in management research: A literature review. *Journal of Management Analytics*, 7(2), 139-172.
- [41] Smeaton, A. F. (1999). Using NLP or NLP resources for information retrieval tasks. In *Natural language information retrieval* (pp. 99-111). Dordrecht: Springer Netherlands.

[42] Wang, X., Wang, H., & Yang, D. (2021). Measure and improve robustness in NLP models: A survey. arXiv preprint arXiv:2112.08313.